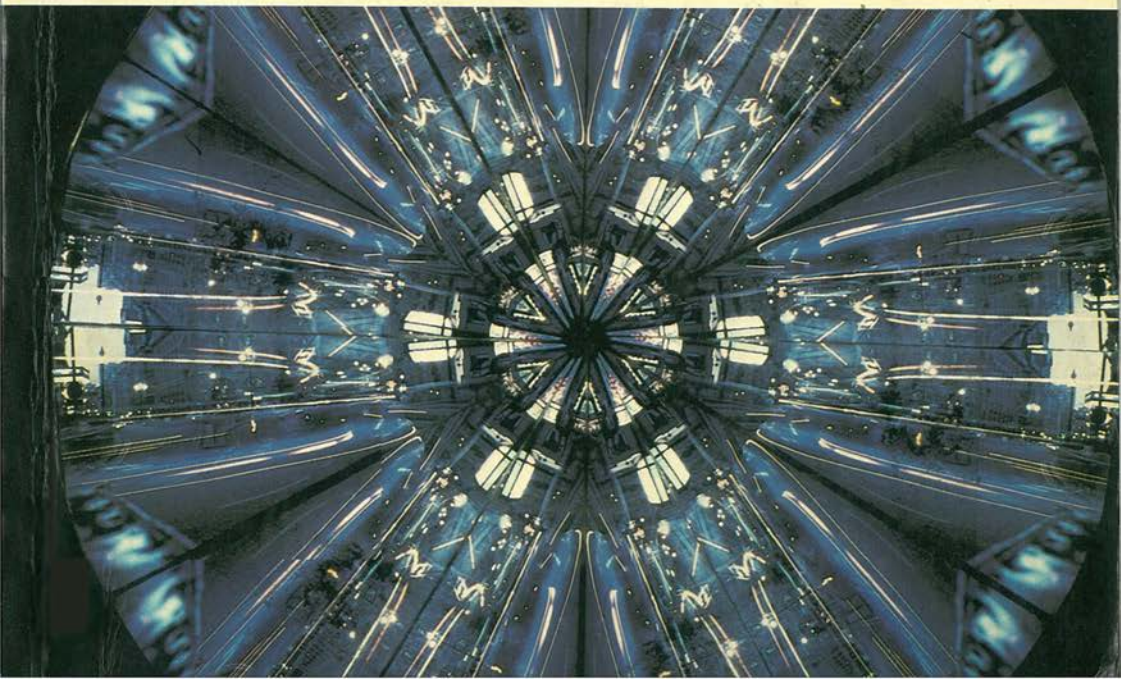
TEA

Un Editor-Assembler Residente per 8080-8085

EDIZIONE ITALIANA

CHRISTOPHER A. TITUS

GRUPPO EDITORIALE JACKSON



TEA

Un Editor-Assembler Residente per 8080-8085

di
Christopher A. Titus

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

Il programma contenuto in questo libro è destinato ad uso proprio del lettore. Non viene concesso nessun diritto, sia diretto che indiretto, alla distribuzione di questo programma in qualsiasi forma, tra le quali audiocassette, floppy disk, qualunque forma di memoria a sola lettura, nastro di carta o qualsiasi altro mezzo.
Le persone interessate ad ottenere tali diritti per la distribuzione di questo software dovrebbero mettersi in contatto con la Tychon, Inc.

Le informazioni contenute in questo libro sono state scrupolosamente controllate.
Tuttavia, non si assumono responsabilità per eventuali errori od omissioni.
È esclusa ogni responsabilità per danni che dovessero derivare dall'utilizzo di questo libro.

© Copyright 1979 per l'edizione americana Christopher A. Titus,

© Copyright 1981 per l'edizione italiana Gruppo Editoriale Jackson

Tutti i diritti sono riservati. Nessuna parte di questo libro può essere riprodotta, riportata in opere simili, posta in sistemi di archiviazione, trasmessa in qualsiasi forma o mezzo meccanico, elettronico, fotocopiatura, ecc. senza l'autorizzazione scritta.

Prima edizione: 1980

Stampato in Italia da:
S.p.A. Alberto Matarelli - Milano - Stabilimento Grafico

Prefazione

Linguaggi evoluti, quali il BASIC e il FORTRAN, benchè siano di impiego assai frequente nella programmazione dei microcalcolatori, non pertanto sono esenti da gravi limitazioni. Essi presentano notevoli difficoltà di adattamento nell'acquisizione a grande velocità dei dati, nei controlli di processo e nell'elaborazione dei dati, oppure con dispositivi periferici di qualche complessità fondati su tecniche di interruzione. Queste ragioni giustificano da sole l'impiego ancora diffuso del linguaggio assembler nella programmazione dei microcalcolatori.

Con il linguaggio assemblatore, ciò che compie in pratica il programmatore è la memorizzazione di istruzioni e di codici corrispondenti ai dati nella memoria del microcomputer. Le istruzioni memorizzate coincidono con le istruzioni che il circuito integrato del microprocessore è in grado di compiere effettivamente in base al modo in cui esso è stato progettato. Nel caso dell'8080 e dell'8085, il *set di istruzioni* di questi due microprocessori è quello stabilito dalla Intel Corporation. Quando si procede alla memorizzazione delle istruzioni e dei valori relativi ai dati, questi sono memorizzati come sequenze di uni e zeri logici. Per molti microprocessori attualmente di larga diffusione, otto stati o livelli logici del genere sono combinati in modo da costituire una *parola* o *byte* di 8 bit. Ciascuna di queste parole o byte di 8 bit è dunque memorizzata in una ben precisa locazione di memoria. Nel caso si dovesse procedere alla programmazione di un microcalcolatore servendosi di parole di 8 bit costituiti da uni e zeri, sarebbe assai difficile scrivere un programma di 200 o 300 istruzioni, per poi memorizzarlo nel microcalcolatore, per il semplice motivo che sarebbe problematico il ricordarsi le corrispondenze tra le istruzioni o i valori dei dati e le varie sequenze di uni e zeri. Nel caso che si programmi un microcalcolatore in questo modo, si dice che esso è programmato servendosi del *linguaggio macchina*.

Anzichè programmare il microcalcolatore in linguaggio macchina, difficile da impiegare e ricordare, si preferisce ricorrere al *linguaggio assemblatore*. Il linguaggio assemblatore corrisponde all'impiego di sequenze brevi di caratteri alfanumerici come MOVAB ed RLC, in qualità di simboli per una particolare sequenza di operazioni eseguibili dal microcalcolatore. L'8080 e l'8085, ad esempio , sono in grado di eseguire l'istruzione "sposta il contenuto di 8 bit del registro B nel registro A ad 8 bit". Sarebbe, in ogni caso, assai difficile scrivere un programma rappresentando ogni istruzione con una descrizione tanto "prolissa". Si preferisce, perciò, valersi di *simboli mnemonici* consistenti, in pratica, in *abbreviazioni* aventi riferimento con la (e) operazione (i) che il microcalcolatore deve eseguire. Il simbolo mnemonico relativo all'istruzione descritta in precedenza è MOVAB. Analogamente il simbolo RLC rappresenta l'istruzione "fai ruotare di un posto a sinistra il contenuto di 8 bit del registro A". Come si può ben immaginare, non è possibile memorizzare nella memoria del microcalcolatore i simboli MOVAB ed RLC e quindi pretendere che il microcalcolatore esegua tali "istruzioni". È, invece, necessario caricare nella memoria del microcalcolatore le istruzioni scritte in linguaggio macchina, 01111000 e 00000111.

Molto opportunamente i fabbricanti di microprocessori forniscono l'elenco di tutti i simboli di istruzione relativi a ciascun microprocessore, unitamente ai codici ad essi corrispondenti in linguaggio macchina. Grazie a ciò è possibile scrivere un programma in forma simbolica, tradurre i simboli di istruzione in linguaggio macchina e porre nella memoria del microcalcolatore gli opportuni codici di linguaggio macchina. È evidente il vantaggio per l'utente, che può in questo modo avere sott'occhio il listing del programma simbolico e rendersi conto di ciò che il microcalcolatore sta per eseguire, invece di dover analizzare una lunga sequenza di uni e zeri.

L'impiego dei simboli di istruzione porta a scrivere il programma in *linguaggio assembler*. Il procedimento, tuttavia, di scrivere (editing) i simboli delle varie istruzioni e poi di tradurli (assembling) nei corrispondenti codici in linguaggio macchina può risultare un lavoro laborioso, monotono e soggetto agli sbagli. Gli *editor* e gli *assembler* sono stati creati appositamente per essere utilizzati nella generazione di programmi in linguaggio assembler per i microcalcolatori.

Un editor non *è* che un programma avente la funzione di manipolare dei caratteri memorizzati in memoria, oppure su cassetta registrata, su nastro perforato o su floppy disk. Di solito, i caratteri ai quali l'editor accede per manipolarli sono posti inizialmente in ingresso al microprocessore da parte di un programmatore che si vale di un'apposita tastiera. I caratteri o sequenze di caratteri possono consistere in simboli di istruzione (ADDX, INXH o CALL) oppure possono addirittura far parte di una lettera (Cari Sherida e Fred). L'editor non ha modo di conoscere, e d'altronde non è affare che lo riguardi, la natura della sequenza di caratteri introdotti. Comunque, una volta che i caratteri sono stati introdotti in memoria, l'editor permette all'utente di cancel-

lare, inserire, ricercare, scambiare, elencare e riporre qualunque sequenza (stringa) di caratteri.

L'impiego dell'assembler permette di tradurre le *stringhe* di caratteri in codici eseguibili dalla macchina. È evidente che la traduzione avviene solamente per certe stringhe. Mentre, cioè, i simboli MOVAB, RLC, INXH, insieme a molti altri, possono essere tradotti in codici eseguibili dalla macchina, non può essere tradotto "Cari Sherida e Fred". Perciò, nel caso che si chieda all'assemblar di tradurre questo simbolo di istruzione, esso genererà un qualche tipo di messaggio di errore. Anche se l'editor è in grado di manipolare qualunque sequenza di caratteri si desideri, questo non vuol dire che l'assembler sia poi in grado di procedere alla sua traduzione. Esso, in realtà può tradurre unicamente sequenze di caratteri molto particolari, rigorosamente definite. In questo libro ci si propone di illustrare l'impiego del Tychen's Editor-Assembler (editore assemblatore della Tychen (TEA) per generare ed assemblare dei programmi in linguaggio assembler. TEA consiste in un editor-assembler co-residente, per via del fatto che l'editor e l'assembler sono presenti in memoria simultaneamente. Una volta introdotto in memoria per mezzo dell'editor un programma *sorgente*, cioè una sequenza di simboli di istruzione, si può passare il controllo all'assembler in modo da generare una versione del programma in linguaggio macchina.

Il numero dei libri appartenenti alla "Blacksburg Continuing Education Series" (collana di Blacksburg per l'educazione permanente) si incrementa rapidamente. I titoli dei libri attualmente disponibili sono elencati in seconda pagina di copertina. Tra i testi in preparazione ve ne sono alcuni dedicati alla programmazione e interfacciamento del 6800, del 6502 e dello Z80. L'editore è sempre interessato a stabilire dei contatti e a collaborare con autori che ritengano che i concetti da essi sviluppati nelle loro opere siano in linea con il taglio di questa collana. Se chi legge coltiva degli interessi che gli farebbe piacere sviluppare, è pregato di mettersi in contatto con il centro di Blacksburg.

Questi testi hanno riscosso notevole successo e raggiunto una vasta diffusione in tutto il mondo sia in classi di tipo tradizionale che presso singoli utenti. Una scelta di essi è in corso di traduzione in tedesco, spagnolo, giapponese, francese, italiano, cinese e malese. Gli interessati ad ottenere maggiori particolari al riguardo di queste traduzioni oppure a tradurre i libri in altre lingue, sono vivamente pregati di rivolgersi a

CHRISTOPHER A. TITUS,
"The Blacksburg Group" (gruppo di Blacksburg)

Prefazione all'edizione italiana

Quanti utilizzano oggi un microprocessore 8080 o 8085 nelle proprie applicazioni? Molti. Quanti hanno a disposizione strumenti software per la compilazione dei programmi? Pochi.
Il programma TEA, il cui listing è riportato nell'ultima appendice di questo volume è senza dubbio un valido contributo per tutti coloro che operano con i suddetti microprocessori.
Esso infatti, una volta caricato su un supporto di memoria, dà la possibilità di scrivere e modificare programmi sorgente scritti in assembler secondo i codici mnemonici di questi due microprocessori.
Oltre a ciò si utilizza in pieno la caratteristica fondamentale del linguaggio assembler, ossia di essere un linguaggio di tipo uno ad uno, in quanto a ogni istruzione simbolica corrisponde una istruzione in linguaggio macchina.
Questo significa un'alta velocità di esecuzione dei programmi, che molto spesso è richiesta nelle applicazioni con microprocessori.
Ci auguriamo che questo volume contribuisca ad aumentare la produzione di software ed a supportare quei tecnici software di microprocessori sempre più necessari in un crescendo sviluppo delle applicazioni di sistemi utilizzanti microprocessori.

Sommario

CAPITOLO 1

MODALITA' DI IMPIEGO DELL'EDITOR 9

Introduzione - Memoria occorrente - L'Editor - Struttura delle Istruzioni - Istruzioni relative al Controllo - Comandi che alterano il Text Buffer - Comandi per accedere al Buffer - Dispositivi di I/O - Considerazioni conclusive sull'Editor

CAPITOLO 2

MODALITA' DI IMPIEGO DELL'ASSEMBLER 51

Caratteristiche dell'Assembler - Frasi (Statements) - Istruzioni valide e loro formato - Pseudo-Op (pseudo-operazioni) - Caratteri Speciali - Assemblaggio dei Programmi - Errori di Assemblaggio - - Assemblaggio di un Programma Memorizzato in memoria R/W - Assemblaggio di un Programma Memorizzato su di un Dispositivo di I/O Perforatore (IOP) - Caricamento in Memoria di un Programma Oggetto - Riferimenti - Bibliografia.

CAPITOLO 3

CONVENZIONI RELATIVE AI DISPOSITIVI DI I/O 107

Caratteristiche delle Subroutine di I/O - Modifica della Tabella di Salto di I/O - Limiti del Dispositivo di I/O

APPENDICE A

FORMATO DELLA BANDA PER IL PROGRAMMA TEA .. 119

APPENDICE B

MODIFICA DEL COMANDO Z 123

APPENDICE C

AGGIUNTA A TEA DI UN COMANDO SPECIALE 125

APPENDICE D

ELENCO RIASSUNTIVO DEI COMANDI DI TEA 127

APPENDICE E

LISTING DI TEA 131

CAPITOLO 1

Modalità di impiego dell'Editor

INTRODUZIONE

La funzione dell'Editor - Assembler Tychen (Tychen Editor - Assembler TEA) è quella di creare, modificare ed assemblare programmi simbolici (sorgenti). Per introdurre in memoria un programma simbolico (cioè una sequenza di simboli mnemonici) ci si può valere di una tastiera ASCII, di un lettore di banda perforata o di un registratore a cassetta. Una volta introdotto il programma, l'impiego di TEA permette di aggiungere, cancellare o modificare linee di testo contenute nel programma simbolico. Per ottenere tale risultato, l'utente non ha che da introdurre un comando di una sola lettera, seguito da uno oppure due numeri di linea; in alternativa, si può introdurre in primo luogo il comando e, quindi, premere il tasto RETURN.

Il programma destinato ad essere assemblato da TEA può risiedere nella sua totalità in memoria oppure può essere spezzato in blocchi diversi memorizzati su nastro. Non esiste virtualmente alcun limite alla dimensione dei programmi che si possono assemblare grazie a TEA. È, cioè, possibile introdurre 100 o 200 linee del programma sorgente procedere all'editing delle relative linee e quindi riporle su nastro perforato. A questo punto tutto il testo deve essere cancellato in modo che sia possibile procedere all'introduzione e all'editing delle successive 100 o 200 linee di testo. È opportuno che la lunghezza di tali blocchi sia mantenuta tra le 100 e le 200 linee, in quanto, così facendo, la correzione degli errori di un blocco si riduce alla lettura di una banda abbastanza breve, contrariamente al caso che il programma sia stato spezzato in lunghi blocchi di 500 o 600 linee; blocchi più estesi significano, infatti, tempi più lunghi di letture, editing e perforazione della versione corretta.

Ogni volta che l'utente preme il tasto RETURN, avviene l'introduzione di una linea di testo nel text buffer (o buffer del testo). La prima linea di testo è la linea 1. Mano a mano che avviene l'introduzione del

testo, TEA provvede automaticamente a generare i numeri di linea, stampandoli in uscita prima che l'utente possa introdurre del testo. Ne consegue che, mentre si procede all'introduzione del testo, non vi è *mai* alcun bisogno di battere un numero di linea da far precedere alla linea di testo. Anche nel caso che si desiderino introdurre o cancellare linee di testo nel corso della procedura di editing TEA procederà automaticamente a cambiare la numerazione di tutte le linee. Poichè la massima capacità del text buffer in un qualsiasi momento è pari a 9999 linee di testo, qualunque riferimento ad una linea di testo prevede il ricorso ad un numero di linea compreso tra 1 e 9999. È d'altronde improbabile che vi siano utenti la cui dotazione di memoria R/W (read/write: a lettura-/scrittura) sia tale da permettere loro di memorizzare testi con più di 9999 linee.

La porzione assembler di TEA ha la funzione di convertire programmi sorgente (espressioni mnemoniche) in programmi oggetto (valori binari propri del linguaggio macchina). Un programma oggetto non consiste in altro che nella versione 'assemblata' del programma stesso ed è suscettibile di *esecuzione diretta* da parte del microcalcolatore 8080. Nel corso del procedimento di assemblaggio, l'impiego di TEA consente di perforare un nastro in codice binario (programma oggetto) e di stampare una versione 'assemblata' del programma stesso, completa di indirizzi in memoria e di valori binari corrispondenti alle espressioni mnemoniche. TEA può, inoltre, provvedere all'essemblaggio del programma *caricando la memoria R/W con una sua versione eseguibile direttamente.* Come già detto, TEA può provvedere sia all'assemblaggio del programma sorgente preventivamente caricato in memoria, sia leggerne una linea alla volta da un dispositivo periferico e assemblarla.

TEA consiste in un programma assemblatore (assembler) a tre passate. Esso cioè, esamina tre volte tutti i caratteri contenuti nel buffer del testo oppure tutti i caratteri perforati su di un nastro.

Nel corso della prima passata, assembler di TEA dispone gli indirizzi simbolici come definiti dall'utente in un'apposita tabella degli indirizzi simbolici, unitamente all'indirizzo a 16 bit corrispondente alla loro "posizione" nel programma. Con la seconda passata, tutte le parole ed i byte definiti dall'utente sono collocati in apposite tabelle, mentre le espressioni mnemoniche contenute nel programma sono interpretate ed i loro equivalenti in codice binario sono perforati su banda o trascitti su audiocassetta. Nel corso di questa passata, inoltre, TEA permette di caricare automaticamente la memoria R/W con questi medesimi codici binari equivalenti. Perciò l'utente ha la possibilità effettiva di dar corso all'esecuzione del programma già a partire dal completamento della seconda passata. La funzione della terza passata è quella di dar luogo ad un listing 'assemblato' del programma.

Questo consiste in uno stampato linea dopo linea relativo al programma dell'utente, completo di indirizzi, di codice operativo dell'espressione mnemonica contenuta nella linea di testo, nonchè commenti vari redatti dall'utente. Il listing relativo alla terza passata può essere ottenuto sia in ottale che in esadecimale, a seconda del sistema di

numerazione preferito. La conversione di TEA dal modo ottale a quello esadecimale, e viceversa, non presenta, infatti, nessuna difficoltà.

Un'altra importante caratteristica di TEA è quella di essere indipendente dal dispositivo di I/O. Da questo dipende l'estrema facilità con cui TEA può essere modificato in modo da comunicare con un registratore per audiocassette anzichè con un lettore/perforatore di banda. In accordo con l'aggiunta o l'eliminazione di dispositivi di I/O dal sistema microcalcolatore basato sull'8080, è facile modificare TEA per adattarlo a questi dispositivi nuovi o semplicemente diversi. Anche nel caso in cui TEA risieda in ROM, la conversione da un dispositivo di I/O all'altro è soltanto questione di pochi secondi.

MEMORIA OCCORRENTE

Lo spazio complessivo di memoria occupato da TEA consiste in poco più di 6K di memoria lettura/scrittura (read/write: R/W) ovvero di memoria programmable a sola lettura (programmabile read-only memory: PROM). Inoltre, per lo stack, una memoria temporanea ed alcuni buffer è necessaria una sezione di memoria R/W per complessive 384 locazioni, mentre la restante parte di memoria R/W può essere utilizzata per la memorizzazione del testo. Allorchè si dà il via iniziale a TEA, esso fa richiesta all'utente degli indirizzi iniziale e finali utilizzabili dal text buffer. Tali indirizzi "limite" possono essere cambiati in qualunque momento per creare lo spazio occorrente a programmi di lunghezza differente. Nella pratica, il cambiamento dei limiti in memoria permette di memorizzare simultaneamente in memoria R/W tre, quattro o più programmi indipendenti. Considerando le dimensioni di TEA e della memoria scratch-pad necessaria, l'impiego di TEA per editare ed assemblare programmi con un qualche dispositivo di I/O tipo una telescrivente un terminale a raggi catodici (cathode-ray-terminal: CRT) o un'audio cassetta rende necessario il disporre di almeno 8K di memoria. L'indirizzo nominale di partenza di TEA è 000 000 (0000 esadecimale).

L'EDITOR

TEA è composto fondamentalmente da due programmi indipendenti, un editor simbolico e un assembler. L'editor interno a TEA permette all'utente di trattare caratteri singoli e stringhe di caratteri presenti nel text buffer. Il completo corredo delle istruzioni a disposizione permettono di utilizzare l'editor per aggiungere, elencare, perforare, investigare, interscambiare, inserire, cancellare o apportare variazioni a linee di testo contenute nel text buffer. È questo un *editor orientato alla linea* (line-oriented), in quanto l'unità minima di testo su cui esso normalmente opera è per l'appunto una linea di testo completa. È, tuttavia, sempre possibile ricorrere all'istruzione di scambio con carattere (stringa) per cambiare una sola lettera nel bel mezzo di una linea di testo. Se si fosse utilizzato un *editor orientato al carattere*, la variazione di caratteri singoli o di stringa isolate di caratteri sarebbe quanto mai

agevole. Tali editor orientati al carattere presentano, tuttavia, l'inconveniente di necessitare di un CRT indirizzabile mediante cursore. La nostra scelta di incorporare in TEA un editor orientato alla linea dipende dal fatto che i terminali di questo genere sono molto meno comuni delle semplici telescriventi seriali asincrone o dei normali CRT. Mentre questo capitolo è dedicato al funzionamento dell'editor, nel Capitolo 2 si tratterrà dell'assembler facente parte di TEA.

STRUTTURA DELLE ISTRUZIONI

Le possibili istruzioni relative a TEA sono di tre tipi. Esse (1) controllano il metodo secondo cui TEA deve operare, (2) rendono possibili i cambiamenti del text buffer e (3) consentono di accedere ad esso senza che intervenga alcun cambiamento. Le istruzioni più semplici sono quelle che si riferiscono al controllo. Esse sono sei e sono descritte nei paragrafi che seguono.

ISTRUZIONI RELATIVE AL CONTROLLO

1. La lettera O è il simbolo impiegato per il comando OCTAL. Quando TEA risponde con C=, è sufficente premere il tasto O per portare TEA nel modo di funzionamento OCTAL. TEA risponderà a questo punto con C=, per cui l'utente può introdurre il comando successivo. Il fatto di porre TEA nel modo OCTAL fa sì che il listing prodotto nel corso della terza passata dell'assembler contenga indirizzi e dati informativi in forma ottale prescindendo dal genere dei numeri utilizzati in un programma.
Quando si introduce il comando Q (QUERY), i limiti della memoria sono stampati in uscita sotto forma di relativi indirizzi di memoria in ottale. Di seguito è riportato il formato relativo a questo comando.

C = O Per introdurre questo comando non vi è bisogno di premere il tasto RETURN (↓).

2. La lettera H è il simbolo impiegato per il comando HEXADECIMAL (esadecimale). Quando TEA risponde con C=, è sufficiente premere il tasto H per porre TEA nel modo di funzionamento HEXADECIMAL.
TEA risponderà a questo punto con C=, per cui l'utente potrà introdurre il proprio comando successivo. Il fatto di porre TEA nel modo HEXADECIMAL fa sì che il listing prodotto nel corso della terza passata dell'assembler ed i limiti della memoria stampati in base la comando Q (Query) saranno espressi con numeri esadecimali.

C = H Per introdurre questo comando non vi è bisogno di premere il tasto RETURN (↓).

3. La lettera M è il simbolo impiegato per il comando MEMORY.

Quando TEA risponde con C=, è sufficiente premere il tasto M perchè TEA permetta all'utente di introdurre nuovi limiti iniziale e finale per gli indirizzi di memoria adibita a text buffer. Questo permette all'utente di maggiorare le dimensioni del buffer oppure di stabilirne uno nuovo da destinare ad un secondo programma ovvero ad un programma aggiuntivo. Quando TEA risponde con INITIAL ADDRESS?, si preme il tasto O (per ottale) oppure quello H (per esadecimale) seguito rispettivamente da un numero ottale di sei cifre oppure da uno esadecimale di quattro cifre, che sarà assunto quale indirizzo iniziale del buffer. Nel caso che siano introdotti dei caratteri ottali o esadecimali non validi, il programma si limita a non prenderli in considerazione.

Si faccia l'ipotesi che si desideri utilizzare TEA per generare un programma da spedire ad un monitor di sistema oppure ad un debugger per la sua esecuzione effettiva. Una volta eseguito il programma, si voglia ripetere TEA una seconda volta allo scopo di correggere gli errori eventualmente contenuti nel programma. Quando TEA è fatto ripartire da 000 000, esso farà nuovamente richiesta degli indirizzi iniziale e finale del text buffer, ma l'utente non avrà alcun bisogno di ricordare gli indirizzi precedentemente introdotti. Sarà infatti sufficiente, al momento in cui TEA pone la domanda relativa a tali indirizzi iniziale e finale, premere il tasto RETURN. TEA utilizzerà allora gli indirizzi precedentemente introdotti e chiederà se esiste per caso una sorgente in memoria, al che l'utente risponderà con Y per YES. Di ciò si tratterà più a fondo nella sezione successiva di questo capitolo.

Allorchè si ricorre al comando M oppure si fissano inizialmente i confini del text buffer al momento di far partire TEA, quest'ultimo utilizza automaticamente la prima locazione di memoria riservata al buffer. Se, ad esempio l'indirizzo iniziale è 040 000 (2000 esadecimale), TEA si servirà di questa locazione di memoria.

La successiva locazione di memoria, 040 001 (2001 esadecimale), è, perciò la prima locazione di memoria utilizzabile per la memorizzazione del testo.

C = M Dopo aver premuto il tasto M non vi è alcun bisogno di premere il tasto RETURN (↵).

4. La lettera R è il simbolo impiegato per il comando READER (lettore). Quando TEA risponde con C=, è sufficiente premere il tasto R perchè avvenga l'introduzione del testo nel buffer destinato ad esso con provenienza dal dispositivo di I/O designato come dispositivo di lettura, quale esempio, un'audio-cassetta, un lettore di nastro, ovvero un floppy disk. Una volta abilitato il dispositivo di I/O con il comando R, l'introduzione del testo nel suo buffer avviene con le modalità disponibili in base ai comandi A (APPEND: assegna), I (INSERT: inserisci) ovvero C (CHANGE: cambia), le cui caratteristiche saranno descritte tra breve. *Il dispositivo di lettura sarà disabilitato soltanto quando alla fine del nastro o dell'audio-cassetta si leggerà CTRL/C.* Una volta disabilitato quest'ultimo, il testo può essere introdotto dalla telescrivente o dal CRT, i dispositivi di I/O maggiormente utilizzati per l'ingresso del testo.

C = R Per dar luogo all'esecuzione del comando non è necessario premere il tasto RETURN (↓) dopo aver premuto quello R.

5. La lettera B è il simbolo impiegato per il comando BOOTSTRAP. La funzione di tale comando è quella di far leggere in memoria la versione 'assemblata' (linguaggio macchina; oggetto) del programma in vista della sua esecuzione. Se, durante il procedimento di assemblaggio il caricamento della versione oggetto in memoria avviene *automaticamente*, il ricorso a questo comando non ha più ragione d'essere.

C = B Dopo aver premuto il tasto B non vi è alcun bisogno di premere il tasto RETURN (↓).

6. La lettera Z è il simbolo impiegato per il comando EXIT, comando precisato dall'utente. Quando, infatti, si preme il tasto Z, il microprocessore 8080, oppure 8085, effettua un salto ad un altro programma già caricato in memoria. Una delle funzioni di questo comando è, perciò, quella di uscire verso un monitor di sistema o un debugger.

C = Z Dopo aver premuto il tasto Z non vi è alcun bisogno di premere di seguito il tasto RETURN (↓).

COMANDI CHE ALTERANO IL TEXT BUFFER

I comandi utilizzabili per alterare il contenuto del text buffer sono soltanto cinque. Questa sezione tratta, per l'appunto, dei quattro comandi più semplici e di uso più frequente. L'impiego di tali comandi permette di compiere delle aggiunte al termine del text buffer (A; APPEND: aggiungere), inserire del testo di fronte ad una certa linea di testo (I; INSERT: inserire), cambiare una linea di testo (C; CHANGE: cambiare) e cancellare una o più linee di testo (D; DELETE: cancellare).

Nel corso di queste sezione del capitolo saranno trattati i vari comandi che determinano la possibilità di alterare il contenuto del text buffer. TEA, perciò, sarà visto operare sia nel *modo* di *comando* che nel *modo* di *testo*. Nel primo di questi due modi, i vari comandi possono essere introdotti in TEA e quindi eseguiti con le modalità già osservate nella sezione precedente. Compiuta l'esecuzione del comando, TEA ritorna al modo di comando per permetterne l'introduzione di uno nuovo comando, tenendo, tuttavia, conto che molti comandi presi in considerazione fan si che TEA passi al modo di testo. Secondo questa procedura, da un dispositivo di I/O si possono introdurre dei caratteri nel text buffer e, in tale sede, questi medesimi caratteri possono essere modificati o cancellati. Il fatto importante è che i caratteri *non* sono interpretati come comandi, poichè TEA non ha alcuna nozione su ciò che è introdotto nel buffer, e d'altronde non è affar suo. Soltanto l'introduzione di

un CTRL/C fa sì che TEA esca dal modo di testo e torni a quello di comando in modo che nuovi comandi possano essere introdotti e le opportune azioni eseguite. Quando si introduce un CTRL/C, TEA risponde con un ?, C =, al che un nuovo comando può essere introdotto.

Append

7. La lettera A costituisce il comando APPEND (aggiungere). Quando TEA risponde con C =, è sufficiente premere il tasto A per portare TEA nel modo di testo secondo cui al testo presente nell'apposito buffer può essere posto in appendice dell'altro testo. Il formato del comando A (APPEND) è riprodotto di seguito.

C = A Dopo aver premuto il tasto A non vi è necessità di premere di seguito il tasto RETURN (↓). TEA passerà comunque nel modo operativo di testo e vi rimarrà sintantochè non sia introdotto un CTRL/C.

La funzione di questo comando è quella di creare un buffer iniziale nonchè di compiere aggiunte di testo al termine del text buffer.

Quando TEA è avviato inizialmente, esso pone tre domande: "INDIRIZZO INIZIALE", "INDIRIZZO FINALE" e "SORGENTE PRESENTE IN MEMORIA?". Ciò è appunto illustrato nell'Esempio 1-1.

Esempio 1-1. Impiego del comando A (APPEND) per la creazione di un buffer iniziale per il testo o per l'aggiunta di altro testo al text buffer.

```
INITIAL ADDRESS ? 0050 000
FINAL ADDRESS ? 0070 000
SOURCE CURRENTLY IN MEMORY ? N
C=A
0001  /THIS IS A TEST
0002
0003          *030 000
0004  LOOP,   JMP
0005          LOOP
0006          0
0007  ?
C=
```

```
INITIAL ADDRESS ?
FINAL ADDRESS ?
SOURCE CURRENTLY IN MEMORY ? Y
C=A
0007  /COMMENT AT THE END
0008  ?
C=L
0001  /THIS IS A TEST
0002
0003          *030 000
0004  LOOP,   JMP
0005          LOOP
0006          0
0007  /COMMENT AT THE END
0008
C=
```

In questo esempio l'utente ha introdotto il numero ottale 050 000 come un indirizzo iniziale per il text buffer, seguito da un indirizzo finale 070 000 ottale. Si osservi come davanti ai due indirizzi si sia fatto uso della lettera 0. TEA stampa uno spazio dopo il primo numero ottale di tre cifre, risparmiando in tal modo all'utente il pensiero di provvedervi personalmente. A questo punto è posta la domanda "SOURCE CURRENTLY IN MEMORY?" (sorgente presente in memoria?). Poichè TEA è stato appena avviato, il buffer non può che essere vuoto, per cui si introduce una N per la risposta NO. Si batte allora il comando A (APPEND), al che, non essendovi testo nel buffer del testo, TEA risponde stampando 0001 come numero di linea. Si introduce allora il comando "/THIS IS A TEST" (questo è un test) seguito da un ritorno del carrello. Il ritorno del carrello fa sì che sia stampato il successivo numero di linea 0002 in vista dell'introduzione di un'altra linea di testo. Una volta introdotte tutte le sei linee di testo, TEA risponde con il numero di linea 0007, ma, poichè il testo è finito, l'utente risponde con un CTRL/C e TEA torna nel modo operativo di comando.

A questo punto l'8080 è RESETTATO e l'utente ha arrestato l'esecuzione di TEA. Lo scopo di tutto ciò è quello di permettere l'esecuzione di un'altro programma o di un'altra operazione. Allorchè l'utente avrà ridato il via a TEA, questo risponderà con le tre domande già viste. In questo caso saranno utilizzati i medesimi indirizzi iniziali e finali in quanto l'utente avrà scelto di rispondere alle domande poste dal programma premendo il tasto RETURN. In ogni caso la risposta alla terza domanda è Y con il significato di YES, dal momento che vi è già stata l'introduzione di un programma. Questa volta, ricorrendo al comando A (APPEND) risulta stampato il numero di linea 0007 in quanto è questa la linea terminale del buffer del testo. Di nuovo, una volta introdotto il commento "COMMENT AT THE END" (commento al termine) e

Esempio 1-2. Utilizzazione del dispositivo di lettura per porre in ingresso del testo nell'apposito buffer con il ricorso al comando A (APPEND).

```
INITIAL ADDRESS ? H2800
FINAL ADDRESS ? H3200
SOURCE CURRENTLY IN MEMORY ? N
C=R
C=A
?
C=L
0001  /THIS IS A TEST
0002
0003          *030 000
0004  LOOP,   JMP
0005          LOOP
0006          0
0007  /COMMENT AT THE END
0008
C=A
0008  /THIS WAS ENTERED FROM THE KEYBOARD
0009  ?
C=L7-8
0007  /COMMENT AT THE END
0008  /THIS WAS ENTERED FROM THE KEYBOARD
C=
```

premuto il tasto RETURN, TEA avrà risposto con il prossimo numero della linea seguente, cioè 0008. Mancando del testo aggiuntivo si introduce un CTRL/C per riportare TEA al modo di comando. Ritorno nel modo di comando TEA riceve in base al comando L (seguito da un RETURN) l'istruzione di procedere al listing di tutto il contenuto del text buffer. TEA procede così ad elencare tutte e otto le linee del testo. Si tralasci per ora di soffermarsi sul formato (struttura) del comando L (LIST: elenca). Il ricorso ad esso è puramente accidentale, dato che l'obiettivo principale dell'esempio è quello di illustrare l'effetto del comando A (APPEND).

Come già detto in precedenza, l'impiego del comando R può essere abbinato a quello del comando A (APPEND) allo scopo di introdurre il testo nel buffer ad esso riservato valendosi di un lettore di nastro o di un'audio-cassetta. Al proposito si veda l'Esempio 1-2.

Aggiunta (Appending) con il Comando R

L'Esempio 1-2 prevedeva che TEA fosse avviato da "scratch" cosicchè non vi era alcun buffer iniziale del testo. Una volta introdotti gli indirizzi iniziale e finale, questa volta in esadecimale (si noti l'impiego dell'H), l'utente risponde con N (NO) alla domanda "SOURCE CURRENTLY IN MEMORY?" (sorgente presente in memoria?). Il dispositivo di lettura, a questo punto è abilitato per introdurre il comando R, mentre avviene l'introduzione del comando A in modo che il testo letto dal lettore di banda o dall'audiocassetta sia aggiunto al buffer del testo già esistente, che inizialmente non contiene nulla.

Una volta letta in memoria tutta la banda, TEA risponde con ?, C =. Si introduce a questo punto il comando L (LIST) in modo da poter 'vedere' ciò che è stato letto nel text buffer dal dispositivo "di lettura". Si noti che ciò che il programma ha letto è esattamente lo stesso testo dell'esempio precedente. Si osservi, inoltre, che il dispositivo di lettura *è disabilitato avvenuta che sia la lettura dell'interno del nastro*. La seconda volta che avviene l'introduzione del comando A (APPEND), il testo è introdotto (/THIS WAS ENTERED FROM THE KEYBOARD: ciò è stato introdotto dalla telescrivente) dalla tastiera della telescrivente, e non dal dispositivo di lettura.

Una volta introdotta questa linea, avviene la stampa di un CTRL/C. TEA riceve quindi l'istruzione di procedere al listing delle sole linee 7 e 8 del text buffer, come appunto ha eseguito.

È importante osservare come, allorchè avviene la lettura di un certo testo da un dispositivo di lettura, il testo stesso non sia stampato su di una stampante o sullo schermo di un CRT. La ragione di questo fatto è che in tal modo il testo può essere letto in memoria alla massima velocità possibile in quanto TEA non è rallentato dal dover stampare delle informazioni su di un dispositivo periferico poco veloce mano a mano che procede la lettura.

Nell'Esempio 1-3 nel microcalcolatore sono introdotti gli indirizzi

Esempio 1-3. Impiego del dispositivo di lettura per aggiungere altro testo al buffer.

```
INITIAL ADDRESS ? H2800
FINAL ADDRESS ? 0065 120
SOURCE CURRENTLY IN MEMORY ? N
C=A
0001  /THESE FIRST FEW LINES WILL BE ENTERED FROM THE
0002  /KEYBOARD.  THEN THE R COMMAND, FOLLOWED BY THE
0003  /A COMMAND, WILL BE USED TO INPUT TEXT FROM THE
0004  /READER, WHICH WILL BE APPENDED TO THE TEXT BUFFER.
0005  ?
C=R
C=A
?
C=L
0001  /THESE FIRST FEW LINES WILL BE ENTERED FROM THE
0002  /KEYBOARD.  THEN THE R COMMAND, FOLLOWED BY THE
0003  /A COMMAND, WILL BE USED TO INPUT TEXT FROM THE
0004  /READER, WHICH WILL BE APPENDED TO THE TEXT BUFFER.
0005  /THIS IS A TEST
0006
0007          *030 000
0008  LOOP,   JMP
0009          LOOP
0010          0
0011  /COMMENT AT THE END
0012
C=
```

iniziale (esadecimale) e finale (ottale), dopodichè la tastiera della telescrivente serve per generare le prime quattro linee di testo ed il lettore di banda per aggiungere al buffer del testo addizionale consistente in sette linee.

Insert (inserisci)

8. La lettera I è quella impiegata per il comando INSERT (inserisci). La funzione di questo comando è quella di permette l'inserimento di una o più linee di testo *a fronte di* un ben determinato numero di linea. Il formato del comando I (INSERT) è riprodotto di seguito.

C = In↓ Inserisci il testo che segue a fronte del numero di linea "n" dopo che sia stato premuto il tasto RETURN (↓).

Se, dopo la I, si introducono un numero di linee *valido* ed un RETURN (↓), TEA passerà nel modo di testo. Esso procederà alla stampa del numero di linea "n", al che l'utente potrà introdurre una o più linee di testo. Nel caso che le linee di testo introdotte siano più d'una, sarà TEA stesso a generare e porre in uscita i numeri di linea che si rendono necessari. In qualunque momento l'utente può introdurre un CTRL/C, ottenendo da TEA la risposta ?, C=, che gli consente di introdurre un nuovo comando (nonchè testo, se necessario).

Analogamente a quanto avviene per il comando A (APPEND), se si introduce un CTRL/C prima che sia introdotta una linea di testo completa (prima cioè che sia stato premuto il tasto RETURN), questa linea non sarà introdotta nel buffer. L'Esempio 1-4 illustra il comando I (INSERT) ed il suo impiego.

Esempio 1-4. Impiego del comando I (INSERT) per aggiungere una linea di testo isolata o più linee al buffer.

```
C=L
0001
0002            *030 000
0003  LOOP,     JMP
0004            LOOP
0005            0
0006
C=I1
0001  /INSERT THIS LINE IN FRONT OF LINE NUMBER 0001
0002  ?
C=L
0001  /INSERT THIS LINE IN FRONT OF LINE NUMBER 0001
0002
0003            *030 000
0004  LOOP,     JMP
0005            LOOP
0006            0
0007
C=I2
0002  /THE INSERT COMMAND CAN ALSO BE USED TO INSERT MULTIPLE
0003  /LINES IN FRONT OF THE SPECIFIED LINE NUMBER.  WATCH HOW
0004  /ALL THE LINES OF TEXT AFTER THESE LINES ARE RE-NUMBERED!
0005  ?
C=L
0001  /INSERT THIS LINE IN FRONT OF LINE NUMBER 0001
0002  /THE INSERT COMMAND CAN ALSO BE USED TO INSERT MULTIPLE
0003  /LINES IN FRONT OF THE SPECIFIED LINE NUMBER.  WATCH HOW
0004  /ALL THE LINES OF TEXT AFTER THESE LINES ARE RE-NUMBERED!
0005
0006            *030 000
0007  LOOP,     JMP
0008            LOOP
0009            0
0010
C=I34
?
C=I
?
C=I3-5
?
C=I7
0007  /THIS WILL NOT BE INSERTED BECAUSE OF A CTRL/C AT ITS END?
C=L
0001  /INSERT THIS LINE IN FRONT OF LINE NUMBER 0001
0002  /THE INSERT COMMAND CAN ALSO BE USED TO INSERT MULTIPLE
0003  /LINES IN FRONT OF THE SPECIFIED LINE NUMBER.  WATCH HOW
0004  /ALL THE LINES OF TEXT AFTER THESE LINES ARE RE-NUMBERED!
0005
0006            *030 000
0007  LOOP,     JMP
0008            LOOP
0009            0
0010
C=
```

Nell'esempio 1-4 si può osservare come l'impiego del comando L (LIST) permette di andare a vedere che cosa già contenga il text buffer. *A fronte della* linea 0001 si inserisce allora una linea di testo (/INSERT THIS LINE IN FRONT OF LINE NUMER 001: / inserisci questa linea a fronte della linea n° 001). Dopo che TEA ha stampato 0002, si introduce un CTRL/C con l'effetto che TEA abbandona il modo di testo per tornare a quello di comando e rispondere con un ?, C=. Per dare un esempio di come nel buffer si inseriscono più linee di testo insieme, si introduce il comando 12.

La risposta di TEA consiste nello stampare 0002 e passare quindi al modo di testo. In questo caso le linee di testo addizionali che si battono sono tre. Si osservi; come, allorchè si introduce un RETURN, TEA provveda come prima cosa a stampare un ritorno del carrello e un avanzamento di linea e, subito dopo, stampi il numero relativo alla successiva linea di testo in sequenza. Dopo che sono state introdotte tutte e tre le linee di testo, TEA stampa infatti 0005. Poichè non vi sono altre linee di testo da inserire, si introduce un CTRL/C e TEA ritorna al modo di comando.

Introducendo il comando L (LIST), si può osservare come TEA provveda a numerare di nuovo le linee di testo in base al numero di linee di testo inserite nel buffer del testo. Terminato il listing, sono stati introdotti alcuni comandi non validi. Il comando 134 è privo di significato in quanto non esiste un numero di linea 34 a fronte del quale si possa inserire del testo. Lo stesso vale per il comando I in questo caso non è specificato il numero di linea a fronte del quale si vuole inserire del testo. Anche il comando 13-5 è da considerarsi non valido dato che vi sono specificati due numeri di linea, mentre il formato del comando I (INSERT) esige che dopo la I e prima di premere il tasto RETURN, sia introdotto un solo numero di linea. Il comando finale 17 sarebbe in sè pienamente valido, ma nel microcalcolatore è stato introdotto un CTRL/C prima che potesse essere premuto il tasto RETURN, con il risultato che la linea 7 non ha potuto essere inserita nel text buffer. È possibile "vedere" che è stato introdotto un CTRL/C dal fatto che TEA ha stampato un ?, seguito da un ritorno del carrello, un avanzamento di linea C=.

Si osservi come per inserire una linea di testo sia sufficiente porre in ingresso a TEA 12 quando esso stampa C=. Non è cioè necessario porre in ingresso 10002 o 1002 anche se, volendo, cioè è sempre possibile. Ciò che si vuol dire è che *dai numeri di linea battuti in TEA si possono eliminare gli zeri iniziali.* I numeri 0002 e 2 sono quindi equivalenti. TEA ignora gli zeri iniziali che sono eventualmente introdotti. Ad esempio in TEA potrebbe essere introdotto il comando 1000000002↓. Anche in questo caso TEA stampa il numero di linea 0002 e quindi si attende che siano inserite una o più linee di testo.

Allorchè TEA inserisce effettivamente nel buffer la linea di testo introdotta dall'utente, esso "spalanca" il buffer del testo per il numero di locazione di memorie volute e quindi scrive la linea di testo nel buffer. È questo il motivo per cui in un programma contenente 600 linee di testo, TEA impiega un pò di tempo per inserire una linea di testo in prossimità della "testa" del buffer. Per meglio comprendere, si esegua TEA nel suo operare. Dopo l'introduzione di un'intera linea di testo. TEA calcola il numero di caratteri effettivamente contenuti in quella linea. Esso sposta allora verso l'alto tutto il testo caricato in memoria dopo il punto di inserimento per tale numero di locazioni di memoria. Nel caso di un programma che contenga 600 linee di testo, può avvenire che 6.000 o 10.000 caratteri debbano essere spostati verso l'alto di 10, 20 o 34 locazioni di memoria a seconda della lunghezza del testo appena intro-

dotto. È chiaro che lo spostamento in memoria di tal numero di caratteri ASCII è un'operazione che richiede un tempo non trascurabile.

Una volta che TEA ha proceduto al loro spostamento ed inserito la linea di testo, esso stampa il prossimo numero di linea in ordine sequenziale per la prossima linea di testo da inserire.

L'impiego di questa tecnica di memorizzazione ad elenco sequenziale evita *ogni* spreco di memoria. Lo scotto da pagare per questa tecnica corrisponde al fatto che l'inserimento effettivo di una linea di testo può comportare un tempo non indifferente (da 0,1 a 0,6 secondi). È d'altronde ben vero che si sarebbe potuto ricorrere ad un'altra tecnica di memorizzazione cioè alla tecnica ad elenco concatenato, ma, con tale tecnica particolare, si verifica uno spreco di memoria non recuperabile a meno di disporre di un programma adeguatamente sofisticato. Gli inserimenti che si ottengono con il metodo ad elenco concatenato sono estremamente rapidi (non si riesce ad individuare nessun ritardo dovuto all'inserimento), ma hanno il difetto di utilizzare male la memoria. Poichè TEA sfrutta tutta la memoria a disposizione, esso esegue le istruzioni di "text collection" nel corso di un inserimento (I; INSERT), un cambiamento (C; CHANGE) ed una cancellazione (D; DELETE).

Cambia

9. La lettera C è quella impiegata per il comando CHANGE. La funzione di tale comando è quella di permettere di cambiare una certa linea di testo (una sola) e, nel caso che si introducano delle linee di testo addizionali, di favorirne l'inserimento nel buffer. Analogamente al comando I (INSERT) è necessario specificare un *ben definito* (unico) numero di linea. Il formato del comando C (CHANGE) è riportato di seguito.

C=Cn ↓ Cambia il numero di linea "n" ed inserisci un qualsiasi numero di linee di testo addizionali dopo la linea "n" una volta che il tasto RETURN (↓) sia stato premuto.

Avvenuta che sia l'introduzione della C, del numero di linea e di un RETURN, TEA passerà al modo di testo purchè il numero di linea sia valido. Avverrà quindi la stampa del numero di linea specificato dal comando dopo di che l'utente può introdurre qualunque testo egli desideri. Tale nuovo testo sostituirà quello vecchio contenuto nel buffer allo specifico numero di linea. Dopo l'introduzione della prima linea di testo, ogni linea supplementare sarà memorizzata nel buffer direttamente di seguito alla prima (modificata). Naturalmente, se sono introdotte delle linee di testo addizionali, TEA provvederà a generare e stampare gli opportuni numeri di linea. Analogamente ai comandi A (APPEND) e I (INSERT), anche in questo caso l'utente può ritornare al modo di comando in qualsiasi momento, bastandogli introdurre un CTRL/C perchè TEA risponda con ?, C=, come preliminare all'introduzione di un nuovo comando. Nel caso che l'introduzione del CTRL/C

Esempio 1-5. Impiego del comando C (CHANGE) per modificare il contenuto del buffer.

```
C=L
0001
0002  /THIS COMENT HIS MANNY MISTEAKS IN IT.
0003
0004          *042 341
0005  LOOPIT, DCRC     /DECREMENT THE COUNT
0006          JNZ      /LOOP BACK IF THE COUNT
0007          LOOPIT   /IS NON-ZERO
0008          0
0009          RET      /WHEN IT IS ZERO, RETURN
0010
C=C2
0002  /CORRECT THE PREVIOUS MISTEAKS, IF POSSIBLE
0003  ?
C=L1-4
0001
0002  /CORRECT THE PREVIOUS MISTEAKS, IF POSSIBLE
0003
0004          *042 341
C=C2
0002  /CORRECT ALL PREVIOUS MISTAKES AND ADD A
0003  /FEW ADDITIONAL LINES OF TEXT TO THE TEXT
0004  /BUFFER.
0005  ?
C=L1-8
0001
0002  /CORRECT ALL PREVIOUS MISTAKES AND ADD A
0003  /FEW ADDITIONAL LINES OF TEXT TO THE TEXT
0004  /BUFFER.
0005
0006          *042 341
0007  LOOPIT, DCRC     /DECREMENT THE COUNT
0008          JNZ      /LOOP BACK IF THE COUNT
```

avvenga prima del RETURN relativo all'ultima linea di testo, questa non sarà introdotta nel buffer.

L'Esempio 1-5 prevede un certo numero di comandi C (CHANGE). Dopo aver proceduto al listing del contenuto di tutto il buffer, si è deciso che la linea 0002 era tanto piena di errori da valer la pena di cambiarla. È stato perciò battuto il comando C2↓. TEA risponde con 0002 e a questo punto si introduce una nuova linea di testo. Allorchè, dopo l'introduzione della E di POSSIBLE, si preme il tasto RETURN, TEA stampa un ritorno del carrello ed un avanzamento di linea, seguiti dal numero di linea della linea successiva. Poichè non vi è altro testo aggiuntivo da introdurre, si batte un CTRL/C per far sì che TEA ritorni al modo di comando. TEA ritorna al modo di comando e stampa un ?, ritorno del carrello, avanzamento di linea C=. L'impiego del comando L (LIST) permette di verificare che la linea 0002 è stata cambiata.

Si è poi deciso non soltanto di cambiare la linea 0002, ma anche di aggiungere due linee di testo addizionali. Si introduce, perciò, nuovamente il comando C2↓. Mentre TEA è nel modo di testo, non solo si cambia la linea 0002 ma si aggiungono altresì le linee 0003 e 0004. Come si poteva prevedere, la linea 0005 (LOOPIT, DCRC/DECREMENT THE COUNT) diviene la linea 0007, il che può essere confermato dall'esame del buffer.

Come già si è avuto occasione di dire, l'introduzione di un CTRL/C

Esempio 1-6. Introduzione in un CTRL/C mentre è in corso l'esecuzione dell'istruzione C (CHANGE).

```
C=C7
0007  LOOPIT, DCRB     /MADE A MISTAKE  (DCRB), TYPE A CTRL/C?
C=L
0001
0002  /CORRECT ALL PREVIOUS MISTAKES AND ADD A
0003  /FEW ADDITIONAL LINES OF TEXT TO THE TEXT
0004  /BUFFER.
0005
0006          *042 341
0007          JNZ      /LOOP BACK IF THE COUNT
0008          LOOPIT   /IS NON-ZERO
0009          0
0010          RET      /WHEN IT IS ZERO, RETURN
0011
C=C34
?
C=C2-4
?
C=C
?
C=
```

prima di un RETURN fa sì che TEA ritorni al modo di comando senza aggiungere la linea di testo al buffer. In ogni caso è bene usare la dovuta cautela nell'introdurre un CTRL/C mentre è in corso l'esecuzione del comando C (CHANGE), come si può notare nell'Esempio 1-6. In tale esempio si è deciso di cambiare la linea 0007, per cui in TEA è stato introdotto come comando C7↓. TEA risponde con il correntto numero di linea a cui fa seguito l'introduzione della linea di testo. Prima tuttavia di premere il tasto RETURN, è stato deciso che la linea di testo conteneva un errore, per cui si è ritornati al modo di comando. Dall'esame del listing del contenuto del buffer si scopre che la linea 0007 è stata cancellata. La ragione di questo fatto è la seguente. Se si introduce un comando C (CHANGE) con un numero di linea valido, la linea di testo così specificata risulta cancellata mentre TEA esegue le stesse istruzioni che sono eseguite allorchè si ricorre al comando I (INSERT). La conseguenza pratica è che il comando C (CHANGE) determina il medesimo risultato che si ottiene con l'esecuzione di un comando D (DELETE) seguito da un comando I (INSERT). L'utente perciò badi particolarmente a valersi correttamente del CTRL/C allorchè TEA sta seguendo il comando C (CHANGE). Questo tipo di problema, naturalmente, viene a mancare se il CTRL/C è introdotto dopo che il tasto RETURN è stato premuto. In tal caso prima avviene l'inserimento nel buffer dell'ultima linea di testo e *solo allora* TEA ritorna al modo di comando. L'Esempio 1-7 rende evidente come il comando C (CHANGE) sia equivalente ad un comando D (DELETE) seguito da un comando I (INSERT). Del comando D (DELETE) si tratterà fra non molto.

Analogamente a quanto avviene con il comando I (INSERT), anche il comando C (CHANGE) fa sì che siano eseguite le istruzioni di TEA relative al text collection. La ragione evidente per cui questo avviene è che il comando C (CHANGE) ha come conseguenza che l'8080 esegua alcune istruzioni identiche a quelle che esso esegue allorchè si introduce il comando I (INSERT). Ne risulta che, allorchè si cambiano delle linee

Esempio 1-7. C (CHANGE) = D (DELETE) + I (INSERT)

```
C=L
0001          *042 341
0002  LOOPIT, DCRC     /DECREMENT THE COUNT
0003          JNZ      /LOOP BACK IF THE COUNT
0004          LOOPIT   /IS NON-ZERO
0005          0
0006          RET      /WHEN IT IS ZERO, RETURN
0007
C=D2

C=L
0001          *042 341
0002          JNZ      /LOOP BACK IF THE COUNT
0003          LOOPIT   /IS NON-ZERO
0004          0
0005          RET      /WHEN IT IS ZERO, RETURN
0006
C=I2
0002  LOOPIT, DCRC     /CHANGE THE COMMENT ON LINE 0002
0003  ?
C=L
0001          *042 341
0002  LOOPIT, DCRC     /CHANGE THE COMMENT ON LINE 0002
0003          JNZ      /LOOP BACK IF THE COUNT
0004          LOOPIT   /IS NON-ZERO
0005          0
0006          RET      /WHEN IT IS ZERO, RETURN
0007
C=C2
0002  LOOPIT, DCRC     /CHANGE 0002 = DELETE 0002, INSERT 0001
0003  ?
C=L
0001          *042 341
0002  LOOPIT, DCRC     /CHANGE 0002 = DELETE 0002, INSERT 0001
0003          JNZ      /LOOP BACK IF THE COUNT
0004          LOOPIT   /IS NON-ZERO
0005          0
0006          RET      /WHEN IT IS ZERO, RETURN
0007
C=
```

all'inizio del buffer, può occorrere un certo tempo perchè TEA esegua tutte le operazioni necessarie alla raccolta del testo e ritorni al punto in cui esso stampa il numero di linea per la prossima linea di testo in successione.

Caratteristiche Comuni dei Comandi A (APPEND), I (INSERT), e C (CHANGE)

Con TEA nel modo di testo, i comandi A (APPEND), I (INSERT) e C (CHANGE) presentano un certo numero di caratteristiche in comune. Già è noto che tutti e tre questi comandi possono essere utilizzati per aggiungere altro testo al buffer e che un CTRL/C "pone fine" al comando (facendo ignorare l'ultima linea di testo nel caso che non sia stato introdotto un RETURN), provocando il ritorno di TEA al modo comune. Oltre a queste vi sono in comune anche delle caratteristiche relative alla correzione degli errori.

Per la correzione di errori nell'*attuale* linea di testo, sintantochè non sia stato premuto il tasto RETURN , si può utilizzare il tasto RUBOUT (cancella). Se, ad esempio, nel buffer è stato battuto TELETYPEWR-TER, si potrebbe utilizzare il tasto RUBOUT per cancellare il TER alla

fine della parola. L'errore sarebbe quindi corretto introducendo ITER. La prima volta che si preme il tasto RUBOUT, si ottiene la stampa di una "\", seguita da una R. Una seconda pressione sul tasto RUBOUT avrebbe come conseguenza la stampa della E, il che significa che anche essa è cancellata dal buffer del testo. Una terza pressione sul tasto RUBOUT determinerebbe la soppressione della T. Ora che il TER è stato cancellato, si può introdurre ITER così che la parola TELETYPEWRITER abbia la sua corretta ortografia. L'Esempio 1-8 illustra l'impiego del tasto RUBOUT per cancellare errori nell'attuale linea di testo.

Esempio 1-8. Impiego del tasto RUBOUT (DELETE) per la correzione di errori nell'attuale linea di testo.

```
C=A
0001   TELETYPEWRTER\RETITER
0002   ?
C=L
0001   TELETYPEWRITER
0002
```

Si supponga che, dopo aver corretto l'ortografia di TELETYPEWRITER, si sia deciso che in realtà nella linea uno si voleva scrivere la parola CATHODE-RAY TUBE. Per far questo si dovrebbe premere un certo numero di volte il tasto RUBOUT o quello DELETE sulla telescrivente o sul CRT. Per cancellare TELETYPEWRITER il tasto dovrebbe essere premuto 14 volte. Cosa accadrà, tuttavia, se il tasto RUBOUT (DELETE) è premuto invece 15 o 20 volte? *Questo non provocherà la soppressione di alcun altro carattere in nessun altra linea di testo.* Una volta che la linea di testo è stata completamente cancellata, TEA si limiterà ad ignorare qualunque altro RUBOUT (DELETE). Soltanto allorchè nella linea di testo sono introdotti dei caratteri, il tasto RUBOUT (DELETE) può essere impiegato per cancellare alcuni ,caratteri (o anche la totalità dei caratteri) di quella linea.

Può accadere, ad un certo punto, che in una data linea di testo, gli errori corretti siano tanto numerosi da rendere difficile la precisione esatta di ciò che è stato corretto e di ciò che non lo è stato. Premendo il tasto LINE-FEED (avanzamento di linea), TEA stamperà un ritorno del carrello ed un avanzamento di linea, seguito dal *medesimo* numero di linea di prima, unitamente a tutti i caratteri contenuti nella singola linea di testo. Questo facilita all'utente il compito di localizzare qualunque errore gli sia sfuggito in precedenza. Una volta che la linea di testo è stata stampata, è possibile compiere su di essa delle aggiunte o delle eliminazioni. Una volta che l'utente è soddisfatto della composizione della linea di testo, questa è introdotta nel buffer premendo il tasto RETURN. L'Esempio 1-9 illustra le capacità di correzione degli errori di TEA.

In questo esempio, come prima cosa si procede a cancellare totalmente il buffer e TEA si accerta che l'utente abbia davvero l'intenzione di eliminare il contenuto dell'intero buffer del testo ponendo la domanda

Esempio 1-9. Impiego del tasto RUBOUT (DELETE) per la correzione di errori e di quello LINE-FEED per osservare i cambiamenti effettuati nell'attuale linea di testo.

```
C=D
ARE YOU SURE ? Y
C=A
0001   TELETYPEWRITER\RETIRWEPYTELETCATHODE-RAY TBUE
0001   CATHODE-RAY TBUE\EUBUBE
0001   CATHODE-RAY TUBE
0002   ?
C=L
0001   CATHODE-RAY TUBE
0002
C=
```

"ARE YOU SURE?" (Sei sicuro?). Poichè la risposta è Y (per YES), ne risulta che l'*intero* buffer è cancellato.

È quindi introdotto il comando A (APPEND) e, poichè il buffer è stato appena cancellato, TEA stampa il numero di linea 0001.

È quindi battuta la parola TYPEWRITER, che però è subito dopo cancellata valendosi del tasto RUBOUT (DELETE). Dopo che la linea è stata interamente cancellata, sono introdotte le parole CATHODE-RAY TUBE. Si preme allora il tasto LINE-FEED in modo che l'utente possa osservare l'esatto contenuto della linea 0001. TEA risponde al LINE-FEED stampando una seconda volta il numero di linea (0001) e quindi i caratteri presenti in quella linea. Allorchè TEA si ferma dopo aver stampato la E di TBUE, l'utente si accorge dell'ortografia errata di TBUE. Il tasto RUBOUT (DELETE), premuto tre volte, cancella i caratteri E, U e B.

Si introducono allora, nell'ordine appropriato, i caratteri corretti. Proprio per assicurarsi che le parole siano scritte correttamente, si preme di nuovo il tasto LINE-FEED, al che TEA risponde nel modo corretto, stampando il numero di linea e le parole CATHODE-RAY TUBE scritte con ortografia corretta. Dato che l'esame è soddisfacente, l'utente introduce nel buffer la linea di testo premendo il tasto RETURN. TEA mette da parte nel buffer la linea e stampa quindi il prossimo numero di linea che segue, 0002. Si introduce allora un CTRL/C in modo che TEA ritorni nel modo di comando. Si introduce infine il comando L in modo che TEA provveda a stampare il contenuto del buffer, unitamente al numero di linea relativo alla linea di testo successiva se questa è stata introdotta.

L'impiego di TEA può anche servire a cancellare interamente l'attuale linea di testo. Questo evita di dover premere 10 o 15 volte il tasto RUBOUT o quello DELETE. Per fare sparire l'intera linea di testo mentre ci si trova nel modo di testo, in conseguenza di uno dei comandi A (APPEND), I (INSERT) o C (CHANGE), è sufficiente introdurre un CTRL/E prima che sia premuto il tasto RETURN. TEA provvederà a cancellare tutti i caratteri presenti nell'attuale linea di testo ed a stampare un ritorno del carrello ed un avanzamento di linea seguiti dallo *stesso* numero di linea. Potrà allora essere introdotta una nuova linea di testo. Analogamente ad un CTRL/C, un CTRL/E non determina la stampa di un carattere specifico. L'impiego del "tasto" CTRL/E è

Esempio 1-10. Impiego del tasto CTRL/E per cancellare l'attuale linea di testo.

```
C=A
0001  /DELETE THIS WITH "RUBOUTS"\"STUOBUR" HTIW SIHT ETELED//LINE 0001
0002  /AND LINE 2.
0003  ?
C=L
0001  /LINE 0001
0002  /AND LINE 2.
0003
C=D
ARE YOU SURE ? Y
C=A
0001  /DELETION WITH A CTRL/E IS EASIER
0001  /LINE 0001
0002  /AND LINE 2.
0003  ?
C=L
0001  /LINE 0001
0002  /AND LINE 2.
0003
C=12
0002  /THIS IS AN INSERTION TEST
0002  /A CTRL/E WAS JUST ENTERED
0003  ?
C=L
0001  /LINE 0001
0002  /A CTRL/E WAS JUST ENTERED
0003  /AND LINE 2.
0004
C=
```

illustrato nell'Esempio 1-10.

Cosa succede nel caso che il buffer si avvicini all'indirizzo finale specificato dall'utente? Anche se dopo tale indirizzo finale vi può essere ancora della memoria disponibile, non è previsto che TEA si serva di questa memoria senza un'esplicita autorizzazione dell'utente. Inoltre occorre, naturalmente, considerare la possibilità al di là dell'indirizzo finale specificato dall'utente *non* vi sia altra memoria. Per evitare che l'utente cerchi di mettere da parte del testo in (1) memoria che non è stata riservata per essere utilizzata da TEA oppure (2) addirittura inesistente, TEA provvede a stampare un messaggio di errore tutte le volte che il buffer del testo giunge ad una distanza inferiore a 256 locazioni di memoria dall'indirizzo finale specificato dall'utente. Nel caso che questo si verifichi, TEA *disabilita* i comandi A (APPEND) e I (INSERT) in modo che in memoria non possa essere introdotto dell'altro testo. Il comando C (CHANGE) non risulta disabilitato.

L'impiego del comando C (CHANGE) è ancora ammesso in quanto esso dapprima cancella una singola linea di testo e quindi attende che l'utente ponga in ingresso una o più linee di testo. Indipendentemente da quanto o da dove TEA giunge internamente alla zona delle 256 locazioni di memoria della fine del buffer, esso lascia che l'utente completi la linea di testo "in contravvenzione", prima di dar luogo alla stampa del messaggio di errore. Ne consegue che l'utente può spingersi entro le ultime 250 o 230 locazioni di memoria alla fine del buffer, senza poter fare altro. Allorchè l'ultima linea di testo è introdotta nel buffer mediante il tasto RETURN, TEA provvede a stampare il messaggio di errore "ME!" (MEMORY EXCEEDED: memoria superata) ritornando quindi al modo di comando.

Esempio 1-11. Esaurimento della memoria.

```
C=M
INITIAL ADDRESS ? 0050 000
FINAL ADDRESS ? 0051 370
SOURCE CURRENTLY IN MEMORY ? Y
C=L
0001  /THIS SUBROUTINE DECREMENTS THE CONTENT
0002  /OF THE C REGISTER UNTIL IT IS ZERO.
0003
0004          *042 341
0005  LOOPIT, DCRC     /DECREMENT THE COUNT IN THE C REGISTER
0006          JNZ      /JUMP BACK TO "LOOPIT" IF THE RESULT
0007          LOOPIT   /IS NON-ZERO
0008          0
0009          RET      /C=000, SO RETURN
0010
C=A
0010  /THIS SHOULD BE THE LAST LINE OF TEXT ALLOWED !!!!!!!!
ME!
C=Q
051 035
051 370
LINE # =  0011
C=A
ME!
C=I3
ME!
C=C10
0010  /THIS COMMENT IS JUST AS LONG AS THE PREVIOUS ONE !!!!
ME!
C=
```

Ad eccezione di A (APPEND) e I (INSERT) possono essere utilizzati tutti gli altri comandi. L'Esempio 1-11 contiene un certo numero di messaggi di errore ME!.

All'inizio dell'Esempio 1-11 si fissa l'indirizzo iniziale a 050 000 ottale e quello finale a 051 370 ottale, con il che l'utente ha riservato 505 locazioni di memoria per il buffer del testo. Poichè in memoria è già contenuto un programma, l'utente lo comunica a TEA e chiede il listing del contenuto del buffer. Al completamento del listing si introduce il comando A (APPEND) in modo che al contenuto del buffer possa essere aggiunto del testo addizionale. Dopo aver introdotto una linea isolata di testo ed aver premuto il tasto RETURN, si introduce la linea di testo ed aver premuto il tasto RETURN, si introduce la line di testo nel buffer. Comunque, dal momento che la fine del buffer si trova ora a meno di 256 locazioni di memoria dall'indirizzo finale, TEA stampa il messaggio di errore ME! e ritorna al modo di comando. Il comando Q non fa che dimostrare che il buffer si trova proprio entro 256 locazioni di memoria a partire dall'indirizzo finale. Nonostante si introduca il comando A (APPEND) TEA stampa il messaggio di errore ME! e ritorna immediatamente al modo di comando, impedendo l'introduzione di testo addizionale. Come previsto, l'utente non ha potuto inserire nemmeno una linea di testo nel buffer (13↓).

Poichè il comando C (CHANGE) può essere ancora impiegato, si introduce un comando C10↓. TEA provvede a cancellare la linea 10 ed a stampare, quindi, il numero 0010. Si introduce a questo punto una nuova linea di testo, che, tuttavia, risulta tanto lunga da far sì che la fine

del buffer si trovi a meno di 256 locazioni di memoria dall'indirizzo finale, cosicchè TEA stampa di nuovo il messaggio di errore ME!.

In questa situazione, i possibili modi per continuare a servirsi dei comandi A (APPEND), e I (INSERT) sono tre. Introducendo il comando M (MEMORY), si ha la possibilità di incrementare l'indirizzo finale, riservando dell'altro spazio di memoria per il buffer. Se, naturalmente, la memoria R/W è stata già sfruttata completamente, questo sistema non potrà essere di aiuto. Un altro modo di risolvere tale problema consiste nel dare il via all'editing del programma, servendosi del comando C (CHANGE) per eliminare da esso i commenti particolarmente lunghi. Generalmente, infatti, i commenti occupano in memoria uno spazio di gran lunga maggiore che non le espressioni mnemoniche e gli indirizzi in forma simbolica. L'ultimo metodo al quale si può ricorrere, nel caso che la memoria a disposizione sia esaurita e che non sia possibili eliminare un buon numero di commenti, è quello di accantonare questa porzione del programma sorgente in qualche dispositivo periferico, quale potrebbe essere un'audio-cassetta, una banda perforata ovvero un floppy disk.

Cancellare

10. La lettera D è quella impiegata per il comando DELETE (cancellare). La funzione di questo comando è quella di permettere di cancellare una linea di testo, un gruppo di linee oppure l'intero buffer. Il formato del comando D è riportato di seguito. La freccia diretta verso il basso (↓) è il simbolo per il tasto RETURN premuto.

C=Dn↓ Cancella la linea con numero "n"
C=Dn-m↓Cancella la linea da "n" a "m"
C=D↓ Cancella l'intero buffer

Indipendentemente dal comando DELETE che si è impiegato, TEA provvede in ogni caso ad apportare gli opportuni cambiamenti dei numeri di linea. Nel caso del primo comando, la linea cancellata è una, cioè la linea (n), cosicchè il numero di linea n+1 sarà tramutato in n, mentre tutti gli altri numeri di linea maggiori di n subiranno un decremento di uno. Allorchè è cancellato un gruppo di linee di testo, TEA provvederà in modo analogo agli opportuni cambiamenti dei numeri di linea.

Se si introduce il comando C=D↓, TEA non procederà a cancellare l'intero buffer sino a che l'utente non abbia risposto alla domanda "ARE YOU SURE?". Tale precauzione evita che l'intero buffer sia cancellato per sbaglio. Rispondendo Y (YES) alla domanda, TEA provvede a stampare un ritorno del carrello seguito da un C= e cancella intanto il buffer. Nel caso che la risposta dell'utente sia N (NO), TEA stampa un punto di domanda, seguito da un ritorno del carrello/avanzamento di linea e C=.

Esempio 1-12. Impiego del comando D (DELETE) per cancellare una linea soltanto, un gruppo di linee e l'intero buffer.

```
INITIAL ADDRESS ? 0050 000
FINAL ADDRESS ? 0070 000
SOURCE CURRENTLY IN MEMORY ? N
C=R
C=A
?
C=L
0001  /THIS IS A TEST
0002
0003          *030 000
0004  LOOP,   JMP
0005          LOOP
0006          0
0007  /COMMENT AT THE END
0008
C=D1
C=L
0001
0002          *030 000
0003  LOOP,   JMP
0004          LOOP
0005          0
0006  /COMMENT AT THE END
0007
C=D3-5
C=L
0001
0002          *030 000
0003  /COMMENT AT THE END
0004
C=D
ARE YOU SURE ? N?
C=L
0001
0002          *030 000
0003  /COMMENT AT THE END
0004
C=D
ARE YOU SURE ? Y
C=L
0001
C=
```

L'Esempio 1-12 illustra il modo di impiego del comando D (DELETE). Dopo aver fissato gli indirizzi iniziale e finale del buffer ed aver comunicato a TEA che al momento non vi è sorgente in memoria, ci si serve del dispositivo di lettura per porre in ingresso del testo nel buffer. Si esegue il comando L (LIST) per esaminare il contenuto del buffer. Una volta completato il listing, si introduce il comando D1↓. Si cancella allora la prima linea contenente il comando "THIS IS A TEST". Effettuando il listing del contenuto del buffer. Si può notare come questa linea sia stata cancellata mentre tutte le linee successive hanno cambiato la loro numerazione. La linea numero 0003, che conteneva * 030 000, ha avuto, dopo la "cancellatura", il nuovo numero 0002 e così tutte le altre.

Si introduce allora il comando D3-5↓, con il risultato di cancellare le linee da 0003 a 0005. Dal listing del contenuto del buffer non è difficile verificare come questa soppressione sia stata eseguita correttamente. Poichè queste linee sono state cancellate, la linea contenente "/COMMENT AT THE END" riceve un nuovo numero, e precisamente il numero di linea 0003. Si osservi come i numeri relativi alla prima a

seconda linea rimangono inalterati.

Si introduce a questo punto il comando per cancellare l'intero buffer del testo (D↓), ma, poichè l'utente ha risposto N (NO) alla domanda posta da TEA, il buffer non è cancellato. Questo può essere verificato mediante il comando L (LIST) che segue il tentativo di eliminazione. L'utente ha cercato allora una seconda volta di cancellare l'intero buffer, rispondendo questa volta alla domanda "ARE YOU SURE?" con Y (YES). L'intero buffer è quindi stato cancellato come appare dal risultato ottenuto allorchè si è proceduto al listing dell'"intero" buffer. Si osservi come TEA stampi il numero per comunicare che, nel caso si introduca una linea di testo utilizzando un altro comando (si potrebbe utilizzare il comando A; APPEND), essa sarebbe automaticamente numerata da TEA come linea numero 0001.

Cosa accade nel caso che si cerchi di cancellare delle linee di testo che non esistono? Dall'Esempio 1-13 appare come TEA non possa cancellare linee di testo inesistenti, cosa che, del resto, era prevedibile. Secondo l'esempio 1-13, nel buffer vi sono le linee da 0001 a 0007, per cui, quando si prova a cancellare delle linee che non esistono, TEA risponde con un punto di domanda, seguito da un C=. Nessuna linea risulta cancellata e TEA si mantiene in attesa del prossimo comando. Si noti anche come TEA possa cancellare la linea 0008. Il buffer conteneva infatti solo le linee da 0001 a 0007 ed il numero 0008 si trova stampato nel listing unicamente per denotare il numero della prossima linea. Si osservi come sia stato fatto anche il tentativo di cancellare le linee da 5 a 12. Nonostante l'esistenza delle linee 5, 6, 7, le linee 8, 9, 10, 11 e 12 non esistono, cosicchè *nessuna* linea risulta cancellata. Ciò è "dimostrato" dal listing in fondo all'esempio.

Esempio 1-13. Tentativo di cancellare linee di testo inesistenti.

```
C=L
0001  /THIS IS A TEST
0002
0003          *030 000
0004  LOOP,    JMP
0005          LOOP
0006          0
0007  /COMMENT AT THE END
0008
C=D34
?
C=D12
?
C=D12-34
?
C=D5-12
?
C=D8
?
C=L
0001  /THIS IS A TEST
0002
0003          *030 000
0004  LOOP,    JMP
0005          LOOP
0006          0
0007  /COMMENT AT THE END
0008
C=
```

Analogamente ai comandi I (INSERT), e C (CHANGE), il comando D (DELETE) esegue anch'esso istruzioni di raccolta del testo. A differenza degli altri due comandi, tuttavia, il comando D (DELETE) riduce le dimensioni del buffer in quanto alcune linee di testo risultano cancellate, mentre i comandi I (INSERT) e C (CHANGE) danno luogo ad un'espansione del buffer. Può dunque capitare che, nel caso che sia cancellato un gran numero di linee nell'ambito di un programma esteso, TEA abbia bisogno di un certo tempo (inferiore comunque al secondo). Se, in un programma di 1000 linee, si cancellano le linee da 100 a 200, le linee da 201 a 1000 debbono essere spostate in memoria verso il basso. Una volta effettuato tale spostamento TEA risponderà con C=, in previsione che sia introdotto un altro comando.

Mentre la conoscenza del meccanismo con cui avviene la raccolta del testo non è affatto essenziale, è però importante rendersi conto che tale operazione si svolge effettivamente e che, allorchè questo avviene, TEA necessiterà di un pur breve lasso di tempo per effettuarla. Tal tempo di attesa può essere notato soltanto se il buffer è particolarmente esteso, costituito da 10.000 o 15.000 caratteri.

Si ricorda che la ragione alla base dell'aver voluto incorporare in TEA la raccolta del testo era quella di poter mettere a disposizione del buffer il maggior spazio di memoria possibile (nei limiti determinati dagli indirizzi iniziale e finale stabiliti dall'utente).

COMANDI PER ACCEDERE AL BUFFER

Ad eccezione di uno, i comandi per accedere al buffer non influiscono sul contenuto di quest'ultimo. Tali comandi permettono all'utente l'esame dei contenuti del buffer, la ricerca di stringhe di caratteri, la conservazione in dispositivi periferici dei contenuti del buffer e l'acquisizione di informazioni riguardo al volume di memoria attualmente impegnata dal buffer ed all'estensione della memoria ad esso riservata. Uno di questi comandi, il comando L (LIST), già utilizzato, è comunque più sofisticato di quanto si possa presumere dagli esempi precedenti.

List (elenca)

11. La lettera L è quella impiegata per il comando LIST. Tale comando può essere utilizzato per 'listare' una linea soltanto un gruppo di linee o l'intero buffer su di un CRT o una telescrivente. Una volta effettuato il listing della (e) linea (e) di testo, TEA ritorna automaticamente al modo di comando. Se, a un dato momento, si vuole arrestare il procedimento di listing e ritornare al modo di comando, è sufficente introdurre un CTRL/C. Il formato per il comando L (LIST) è riprodotto di seguito.

C=Ln↓ Fai il listing della linea numero "n"
C=Ln-m↓ Fai il listing delle linee da "n" a "m"
C=L↓ Fai il listing del contenuto dell'intero buffer

Come si è già potuto osservare nelle occasioni in cui ci si è serviti del comando L (LIST), la linea (o le linee) di testo sono stampate unitamente ai numeri di linea generati da TEA. Questo avviene puntualmente indipendentemente dal comando L utilizzato.

L'Esempio 1-14 illustra per l'appunto l'impiego del comando L (LIST).

In tale esempio, essendo stato introdotto il comando L↓, si ottiene il listing dell'intero buffer. Le linee 0001 e 0005 sono 'listate' individualmente poichè sono stati introdotti i comandi L1↓ ed L5↓.

Esempio 1-14. Impiego del comando L (LIST).

```
C=L
0001            *042 341
0002  LOOPIT,   DCRC     /DECREMENT THE COUNT
0003            JNZ      /LOOP BACK IF THE COUNT
0004            LOOPIT   /IS NON-ZERO
0005            0
0006            RET      /WHEN IT IS ZERO, RETURN
0007
C=L1
0001            *042 341

C=L5
0005            0

C=L2-3
0002  LOOPIT,   DCRC     /DECREMENT THE COUNT
0003            JNZ      /LOOP BACK IF THE COUNT

C=L4-20
0004            LOOPIT   /IS NON-ZERO
0005            0
0006            RET      /WHEN IT IS ZERO, RETURN
0007
C=L15
?
C=L15-20
?
C=L10234
?
C=
```

È stato poi introdotto il comando L4-20↓ (che sembrerebbe un comando non valido per la semplice ragione che non vi sono linee con numeri maggiori di 6). Poichè, tuttavia, è talvolta difficile ricordarsi il numero dell'ultima linea del buffer, nel caso che nel comando di listing il *secondo numero di linea non sia valido*, TEA provvederà comunque al listing del buffer, a partire dal primo numero di linea specificato e continuando sino a che non sia raggiunta la fine del buffer del testo. TEA ritornerà quindi al modo di comando.

Punch (perfora)

12. La lettera P è quella impiegata per il comando PUNCH. La funzione di questo comando consiste nel permettere di conservare l'intero buffer o delle sue porzioni in un dispositivo periferico. Tra i dispositivi di uso più frequente sono compresi un perforatore di banda e un registratore di audio-cassette. L'impiego delle audio-cassette è effettivamente possibile, nonostante il nome del comando "implichi" l'impiego di un perforatore vero e proprio. Una volta perforata la linea (le linee) di testo, TEA ritorna nel modo di comando. Il formato dei tre differenti comandi punch è quello che segue.

C=Pn↓ Metti da parte la linea "n" del dispositivo periferico
C=Pn-m↓ Metti da parte le linee da "n" a "m" nel dispositivo periferico
C=P↓ Metti da parte il contenuto dell'intero buffer nel dispositivo periferico.

Prima che qualsiasi parte di testo sia passata al dispositivo periferico, TEA pone in uscita 128 caratteri (200 in ottale 80 in esadecimale) che formano il leader (la testa) del programma. È quindi posta in uscita la

Esempio 1-15. Impiego del comando P (PUNCH) per accantonare del testo su di un dispositivo periferico.

```
C=L
0001            *042 341
0002  LOOPIT,   DCRC     /DECREMENT THE COUNT
0003            JNZ      /LOOP BACK IF THE COUNT
0004            LOOPIT   /IS NON-ZERO
0005            0
0006            RET      /WHEN IT IS ZERO, RETURN
0007
C=P1
*042 341

C=P3-5
JNZ/LOOP BACK IF THE COUNT
LOOPIT/IS NON-ZERO
0

C=P
*042 341
LOOPIT,DCRC/DECREMENT THE COUNT
JNZ/LOOP BACK IF THE COUNT
LOOPIT/IS NON-ZERO
0
RET/WHEN IT IS ZERO, RETURN

C=P4-15
LOOPIT/IS NON-ZERO
0
RET/WHEN IT IS ZERO, RETURN

C=P34
?
C=P10-30
?
C=
```

parte di testo specificata, seguita da un CTRL/C (203 in ottale, 83 in esadecimale). Dopo il CTRL/C sono posti in uscita 128 (200 in ottale 80 in esadecimale) caratteri di chiusura (trailer). Allorchè i caratteri di chiusura sono stati posti in uscita, TEA ritorna nel modo di comando. La ragione per cui, alla fine del testo, mette da parte il CTRL/C è di far sì che, allorchè il testo è riletto di nuovo in TEA con l'impiego dei comandi R (READER) e A (APPEND), I (INSERT) e C (CHANGE), TEA possa accorgersi di aver letto tutto il testo. Utilizzando come dispositivo periferico una telescrivente equipaggiata con un perforatore di banda si otterrebbe un risultato analogo a quello presentato dall'Esempio 1-15.

Come si può osservare nell'Esempio 1-15, TEA provvede a *comprimere* il testo mano a mano che questo è posto in uscita verso il dispositivo periferico. Mentre non è tanto importante conoscere il meccanismo per giungere a questo risultato, è invece fondamentale il fatto che lo scopo è quello di inviare in uscita al dispositivo periferico il minor numero possibile di caratteri. Come si può notare, il comando P (PUNCH) mette in grado di accantonare una linea, un gruppo di linee ovvero il contenuto dell'intero buffer in un dispositivo periferico.

Analogamente a quanto avviene con il comando L (LIST), se nel comando P (PUNCH) che si introduce, il *secondo numero di linea non è valido*, TEA provvederà lo stesso a perforare il contenuto del buffer, a partire dalla prima linea specificata e continuando sino a che non sia stata raggiunta la fine del buffer. TEA ritornerà quindi al modo di comando, in attesa che sia introdotto un altro comando.

Query

13. La lettera Q è quella impiegata per il comando QUERY. La funzione di tale comando è quella di permettere di determinare l'indirizzo dell'ultima locazione di memoria utilizzata dal buffer, l'indirizzo dell'ultima locazione di memoria riservata al buffer ed il numero di linee attualmente contenute nel buffer. Il formato del comando Q (QUERY) è riportato di seguito.

C=Q Dopo aver premuto il tasto Q, non vi è bisogno di premere il tasto RETURN (↓)

L'Esempio 1-16 presenta alcuni risultati ottenuti con l'impiego del comando Q.

Dopo aver avviato TEA, sono stampate le richieste per un indirizzo iniziale e finale. Dopo ciascuna domanda è premuto il tasto RETURN, in quanto l'utente intende continuare a valersi dei limiti di indirizzo stabiliti in precedenza. Dopo una risposta Y alla domanda "SOURCE CURRENTLY IN MEMORY?" (sorgente presente in memoria?), si passa al listing del programma contenuto nel buffer. Si introduce allora

Esempio 1-16. Impiego del comando Q (QUERY) per determinare l'utilizzazione della memoria ed i suoi limiti.

```
INITIAL ADDRESS ?
FINAL ADDRESS ?
SOURCE CURRENTLY IN MEMORY ? Y
C=L
0001            *042 341
0002  LOOPIT,   DCRC     /DECREMENT THE COUNT
0003            JNZ      /LOOP BACK IF THE COUNT
0004            LOOPIT   /IS NON-ZERO
0005            0
0006            RET      /WHEN IT IS ZERO, RETURN
0007
C=Q
050 200
070 000
LINE # =   0007
C=H
C=Q
28 80
38 00
LINE # =   0007
C=M
INITIAL ADDRESS ? H2800
FINAL ADDRESS ? H3800
SOURCE CURRENTLY IN MEMORY ? Y
C=Q
28 80
38 00
LINE # =   0007
C=O
C=Q
050 200
070 000
LINE # =   0007
C=
```

il comando Q (QUERY) e TEA stampa l'indirizzo 050 200, indirizzo dell'ultima locazione di memoria utilizzata dal buffer, (*non* lo si confonda con l'indirizzo della locazione di memoria dove è memorizzata la N di RETURN). Avviene quindi la stampa dell'indirizzo dell'ultima locazione di memoria riservata al buffer (070 000 ottale), coincidente con l'indirizzo finale specificato in precedenza. L'ultima linea che TEA stampa in seguito a questo comando è "LINE # = 0007". Come si può notare questo non è propriamente il numero delle linee di testo contenute nel buffer bensì è il numero della prossima linea di testo da introdurre, se mai ne esiste una.

Valendosi del comando H (HEXADECIMAL), dopo che TEA ha stampato il "numero di linee", si fa passare TEA al funzionamento con numeri esadecimali. Ciò è evidente la seconda volta che si introduce il comando Q poichè ora gli indirizzi 050 200 e 070 000, sono stampati sotto forma dei loro numeri equivalenti esadecimali; 2880 e 3800. Dopo la stampa di questi due numeri, è stampato anche il numero della prossima linea di testo, nell'ipotesi che ve ne sia una. Si introduce a questo punto il comando M (MEMORY), introducendo in forma di numeri esadecimali i *medesimi* limiti di memoria. Poichè TEA si trova ancora nel modo esadecimale, il comando Q introdotto subito dopo dà luogo alla stampa degli stessi numeri esadecimali di prima. Il comando O (OCTAL) riporta indietro al modo operativo ottale, come si può notare dai risultati stampati in seguito all'ultimo comando Q (QUERY).

Search (ricercare)

14. La lettera S è quella impiegata per il comando SEARCH. La funzione di questo comando è quella di rintracciare nel buffer l'esistenza di una stringa. Il formato per il comando S (SEARCH) è riportato qui di seguito.

C=S=STRING (CTRL/S)

Allorchè tale comando è introdotto, TEA risponde stampando un segno di uguale (=) dopo la S battuta dall'utente. È possibile, a questo punto, introdurre una stringa di lunghezza sino a 10 caratteri. Tale stringa non deve contenere ne ritorni del carrello, ne avanzamenti di linea e nemmeno "tab" (tabulazioni) (delle quali si daranno dei cenni). Una volta introdotta la stringa dei caratteri, occorre introdurre un CTRL/S. Allorchè TEA riceve in ingresso il carattere CTRL/S, comincia la ricerca a partire dall'inizio del buffer. Esso si fermerà nella sua ricerca soltanto (1) quando avrà trovato una stringa di caratteri uguale oppure (2) quando sarà giunto alla fine del buffer. Se TEA trova una stringa "gemella", provvede a stampare il numero di linea della ricerca contenente la stringa, unitamente alla linea di testo completa. TEA ritornerà, quindi, al modo di comando. Nel caso che TEA non si imbatta in una stringa corrispondente, provvederà a stampare un ?, tornando poi al modo di comando.

Se si comincia a battere una stringa e ci si accorge di aver commesso un errore nella stringa stessa, si rimedierà introducendo un CTRL/C. TEA ritornerà così al modo di comando per cui si potrà introdurre di nuovo il comando S (SEARCH) e quindi la stringa voluta. Il comando S (SEARCH) non può essere utilizzato per localizzare una stringa contenuta in parte in una linea e parte in un'altra. Il comando S (SEARCH), cioè non può valicare le linee.

L'Esempio 1-17 contiene una dimostrazione pratica del comando S

Esempio 1-17. Ricerca di stringhe corrispondenti nel buffer.

```
C=L
0001            *042 341
0002  LOOPIT,   DCRC     /DECREMENT THE COUNT
0003            JNZ      /LOOP BACK IF THE COUNT
0004            LOOPIT   /IS NON-ZERO
0005            0
0006            RET      /WHEN IT IS ZERO, RETURN
0007
C=S=LOOPIT
0002  LOOPIT,   DCRC     /DECREMENT THE COUNT

C=S=-ZERO
0004            LOOPIT   /IS NON-ZERO

C=S=RET
0006            RET      /WHEN IT IS ZERO, RETURN

C=S=STRING TOO LONG!?
C=S=TEST1?
C=S=TTYOUT?
C=S=BASIC?
C=
```

(SEARCH) e del modo di impiegarlo. La prima stringa che si vuole rintracciare è "LOOPIT". Si noti che, quando si introduce CTRL/S, nulla è stampato sulla stampante. TEA trova questa stringa nella linea 0002, cosicchè stampa il numero di linea e quindi la linea di testo. Si introduce poi la stringa "ZERO", che TEA trova nella linea 0004. Si osservi come nella stringa che TEA cerca di rintracciare possa essere contenuta una lineetta, come pure qualunque altro carattere di stampa, e non solamente lettere e numeri. La terza stringa che TEA è chiamata a rintracciare è "RET", trovata nella linea 0006. È stata quindi introdotta una stringa contenente 16 caratteri (STRING TOO LONG!). Invece di esplorare il buffer del testo alla ricerca di una stringa corrispondente, TEA si è limitata a stampare un ?, seguito da un C=. Sono poi state introdotte tre altre stringhe, ma, non avendo potuto trovare nel buffer una stringa corrisponde, TEA ha stampato ?, seguita da C=, dopo ciascuna di esse.

Next (successivo)

15. La lettera N è quella impiegata per il comando NEXT. La funzione di questo comando è quella di far si che si possa rintracciare la seconda, la terza, e così via, comparsa della stringa nel buffer. Se TEA fosse dotato del solo comando S (SEARCH), sarebbe infatti impossibile scoprire tali comparse addizionali della stringa. Il formato per il comando N (NEXT) è riportato qui di seguito.

C = N Dopo aver premuto il tasto N non è necessario premere il tasto RETURN (↓).

Allorchè TEA stampa C=, è sufficiente premere il tasto N per dar luogo alla sequenza NEXT di istruzioni contenuta in TEA.

TEA continuerà allora a cercare, cominciando dalla linea che *viene dopo* quella contenente la precedente stringa corrispondente. Nel caso che si trovi un'altra corrispondenza, TEA provvede a stampare il numero di linea e la linea di testo. Se non si trovano ulteriori corrispondenze, TEA si limita a rispondere con ?, C=. L'impiego del comando N (NEXT) può essere studiato nell'Esempio 1-18.

Esempio 1-18. Impiego del comando N (NEXT) per trovare nel buffer presenze ulteriori della stessa stringa.

```
C=S=LOOPIT
0002  LOOPIT, DCRC     /DECREMENT THE COUNT

C=N
0004          LOOPIT   /IS NON-ZERO

C=N?
C=S=ZERO
0004          LOOPIT   /IS NON-ZERO

C=N
0006          RET      /WHEN IT IS ZERO, RETURN

C=N?
```

Secondo questo esempio, si introduce in TEA la stringa di ricerca LOOPIT. TEA risponde stampando il numero di linea (0002) e la linea di testo che contiene la stringa corrispondente. Si introduce a questo punto il comando N e TEA trova un'altra stringa corrispondente nella linea quattro, che provvede a stampare in uscita. Introducendo ancora il comando N (NEXT), TEA stampa un ? e C=, poichè non riesce a trovare nessuna ulteriore stringa corrispondente. Si introduce poi la stringa ZERO e TEA trova una stringa corrispondente nella linea quattro. Si introduce di nuovo il comando N e TEA trova una seconda stringa corrispondente nella linea cinque. Quando si introduce ancora il comando N (NEXT), TEA non può trovare nessun'altra stringa corrispondente.

Il comando N (NEXT) non può servire a trovare la seconda, o comunque ulteriore presenza di una stringa nella medesima linea di testo. Ciò è evidenziato nell'Esempio 1-19. Il programma originale è stato modificato con il semplice ricorso al comando C (CHANGE). La linea due ha subito variazioni in modo da contenere due volte la stessa stringa (DCRC). Si introduce allora il comando S (SEARCH), introducendo dopo il segno di uguaglianza la stringa di ricerca DCRC. TEA trova nella linea due la prima comparsa di DCRC. Il comando N, a questo punto, impartisce a TEA l'ordine di continuare a cercare, a partire dalla linea tre, cosicchè la seconda comparsa di DCRC nella linea due non è stata rilevata.

Esempio 1-19. Impiego del comando N (NEXT) per tentare di trovare la prima ricomparsa della stessa stringa nella stessa linea di testo.

```
C=L
0001            *042 341
0002  LOOPIT,   DCRC     /DCRC THE COUNT
0003            JNZ      /LOOP BACK IF THE COUNT
0004            LOOPIT   /IS NON-ZERO
0005            0
0006            RET      /WHEN IT IS ZERO, RETURN
0007
C=S=DCRC
0002  LOOPIT,   DCRC     /DCRC THE COUNT

C=N?
```

Exchange (Scambiare)

16. La lettera E è quella impiegata per il comando EXCHANGE.

La funzione di tale comando è quella di far sì che la nuova stringa prenda il posto di una vecchia stringa reperita nel buffer mediante i comandi S (SEARCH) o N (NEXT). I metodi in cui il comando E (EXCHANGE) può essere utilizzato sono due e i relativi formati sono riportati qui di seguito.

C=E=NEW STRING (CTRL/E) ... (RETURN non necessario)
C=ES ..... (RETURN non necessario)

Allorchè, mediante i comandi S (SEARCH) o N (NEXT) si trova una certa stringa, si può utilizzare il comando E (EXCHANGE) per sostituire tale stringa con una stringa nuova. Dopo che la stringa è stata trovata, e TEA è ritornato al modo di comando, si introduce il comando E (EXCHANGE) seguito da un segno di uguale. A differenza del comando S (SEARCH), dopo l'introduzione del comando E (EXCHANGE) TEA non provvede da sè a stampare il segno di uguale. Una volta introdotto il segno di uguale, l'utente può introdurre una stringa *di lunghezza a piacere*. Tale nuova stringa sostituisce la presenza di quella vecchia allorchè l'utente introduce un CTRL/E in coda alla nuova stringa. Una volta avvenuto lo scambio delle stringhe, TEA ritorna al modo di comando in attesa che sia introdotto un nuovo comando.

Nel caso che in un programma si debba sostituire un certo numero di stringhe del tipo di LOOPIT oppure TEMPO, è possibile procedere allo scambio delle linee di testo valendosi del comando C (CHANGE) oppure rintracciando le varie stringhe (comandi S o N) e quindi permutando la "vecchia stringa" con una nuova specificata dal comando E (EXCHANGE). Tuttavia questo sistema potrebbe risultare alquanto laborioso se la nuova stringa da battere per il comando E (EXCHANGE) dovesse essere ripetuta, poniamo, 10 o 15 volte. In alternativa, potrebbe risultare conveniente introdurre la nuova stringa *una volta soltanto* e scambiare con essa la vecchia stringa ogni volta che la si scopre nel buffer. Questo è, per l'appunto, possibile se si utilizza il comando ES (EXCHANGE SAME STRING).

Si raccomanda di usare il comando per lo scambio delle stringhe con molta cautela poichè spesso è facile confondersi nel ricordare i vari termini degli scambi effettuati. In pratica, per ridurre le possibilità di errore, *il comando E (EXCHANGE) è impedito da TEA di essere operativo sino a che non si sia trovata una stringa corrispondente. Una volta avvenuto lo scambio di una stringa, non può aver luogo un'altro scambio sino a che non sia localizzata un'altra stringa mediante uno dei due comandi S (SEARCH) o quello N (NEXT).*

Il funzionamento del comando per lo scambio di stringa (E; EXCHANGE) è illustrato nell'Esempio 1-20. Si provvede in primo luogo al listing del contenuto del buffer in modo da conoscere quanto in esso presente. Si ricerca quindi la stringa LOOPIT, per la quale TEA trova una stringa corrispondente nella linea due. Si introduce allora il comando E (EXCHANGE) seguito da un segno di uguale e dalla stringa (TEST3) che deve sostituire LOOPIT. Allorchè, alla fine della nuova stringa (TEST3), si introduce il CTRL/E, il relativo scambio ha luogo, come si può verificare introducendo il comando L2↓. Poichè, a questo punto, è stato erroneamente introdotto una seconda volta il comando E, TEA risponde con ? e C=. Dato che è appena avvenuto uno scambio, il comando E non può essere usato una seconda volta, sino a che nel buffer, non sia stata trovata una seconda stringa.

Con il comando N (NEXT) si impartisce allora a TEA l'ordine di continuare a cercare la stringa LOOPIT, a partire dalla linea tre.

Esempio 1-20. Scambio di stringhe.

```
C=L
0001            *042 341
0002  LOOPIT,   DCRC     /DCRC THE COUNT
0003            JNZ      /LOOP BACK IF THE COUNT
0004            LOOPIT   /IS NON-ZERO
0005            0
0006            RET      /WHEN IT IS ZERO, RETURN
0007
C=S=LOOPIT
0002  LOOPIT,   DCRC     /DCRC THE COUNT

C=E=TEST3
C=L2
0002  TEST3,    DCRC     /DCRC THE COUNT

C=E?
C=N
0004            LOOPIT   /IS NON-ZERO

C=E=SECOND
C=L4
0004            SECOND   /IS NON-ZERO

C=L
0001            *042 341
0002  TEST3,    DCRC     /DCRC THE COUNT
0003            JNZ      /LOOP BACK IF THE COUNT
0004            SECOND   /IS NON-ZERO
0005            0
0006            RET      /WHEN IT IS ZERO, RETURN
0007
C=S=SECOND
0004            SECOND   /IS NON-ZERO

C=E=TEST3
C=S=TEST3
0002  TEST3,    DCRC     /DCRC THE COUNT

C=N
0004            TEST3    /IS NON-ZERO

C=S=TEST3
0002  TEST3,    DCRC     /DCRC THE COUNT

C=E=NEW
C=N
0004            TEST3    /IS NON-ZERO
C=ES
C=L
0001            *042 341
0002  NEW,      DCRC     /DCRC THE COUNT
0003            JNZ      /LOOP BACK IF THE COUNT
0004            NEW      /IS NON-ZERO
0005            0
0006            RET      /WHEN IT IS ZERO, RETURN
0007
C=
```

Questa stringa è rinvenuta nella linea quattro e scambiata, stavolta, con la stringa SECOND, come si può verificare da listing delle linea quattro (L4↓). È evidente che, dopo uno scambio, l'introduzione di un comando di listing non è per niente necessario. Allo scopo di avere un quadro di tutto l'"insieme" si introduce il comando L in modo che sia effettuato il listing dell'intero buffer. Come previsto. LOOPIT è stato rimpiazzato la prima volta da TEST3 e la seconda da SECOND.

Si passa allora alla ricerca della stringa SECOND e la si sostituisce con la stringa TEST3. Cosicchè nel buffer vi sono adesso due stringhe TEST3 (linee due e quattro). Per verificare che TEST3 è ora presente nel buffer due volte, si introduce il comando S=TEST3. TEA trova tale stringa nella linea due e provvede a stamparla in uscita. Il comando N (NEXT) scopre per la seconda presenza della stringa nella linea quattro. Si introduce ancora il comando S=TEST3, ma, questa volta, dopo che TEA trova la stringa, si introduce il comando E=NEW, con il quale si ottiene lo scambio del primo TEST3 con NEW. Il comando N scopre la seconda presenza di TEST3 (nella linea quattro) ed il comando ES fa avvenire lo scambio di TEST3 con NEW (la stringa "nuova" introdotta in precedenza). Il listing finale permette di verificare che entrambe le ricorrenze della stringa TEST3 sono state sostituite con NEW.

Assemble (assemblare)

17. La lettera X è quella impiegata per il comando ASSEMBLE. Dopo tale comando non è necessario introdurre nessun RETURN. Allorchè si introduce questo comando, l'8080 comincia ad eseguire la porzione di assembler di TEA. Per le istruzioni di impiego dell'assembler si consulti il Capitolo 2.

C=X Dopo aver premuto il tasto X non vi è alcun bisogno di premere il tasto RETURN (↓).

Caratteri particolari

Tasto RETURN - Mentre è valido il modo di comando, vi sono molti comandi che non possono essere eseguiti sino a che non sia stato premuto il tasto RETURN. Tali comandi comprendono I (INSERT), D (DELETE), C (CHANGE), L (LIST) e P (PUNCH). Per altri comandi si ha l'esecuzione immediata nel momento stesso in cui sono introdotti, in quanto TEA non attende che sia premuto il tasto RETURN. Tali comandi comprendono A (APPEND), R (READER), H (HEXADECIMAL), O (OCTAL), X (ASSEMBLE), N (NEXT), e Q (QUERY). I comandi S (SEARCH) ed E (EXCHANGE) esigono rispettivamente caratteri terminali particolari CTRL/S e CTRL/E.

Mentre ci si trova in modo di testo, in seguito ad un comando A (APPEND), I (INSERT) e C (CHANGE), una linea di testo non sarà introdotta nel buffer del testo sino a che non sia premuto il tasto RETURN. La risposta di TEA consisterà nel conservare la linea nel buffer del testo e nello stampare il numero di linea della prossima linea di ordine crescente.

Tasto LINE-FEED - Allorchè TEA sta perforando una banda con il contenuto del buffer, ad ogni ritorno del carrello così perforato fa seguito un line-feed. Allorchè la banda è riletta di nuovo in TEA, questi line-feed sono, tuttavia, ignorati. Mentre ci si trova in modo di testo, un

line-feed fa sì che siano stampati lo stesso numero di linea ed il testo contenuto in quella linea. Ciò si dimostra particolarmente utile nel caso che si sia corretto un certo numero di errori valendosi del tasto RUBOUT.

Per ulteriori informazioni si consulti la sezione intitolata *Caratteristiche Comuni dei Comandi A (APPEND), I (INSERT), e C (CHANGE)* nelle prima parte di questo Capitolo.

Tasto RUBOUT - La funzione di questo tasto, come si è già avuto occasione di dire, è quella di permettere la correzione di errori presenti nelle linee di testo. Il tasto può essere impiegato solamente per correggere errori in una linea di testo *al momento in cui essa è battuta in ingresso*. Non appena il tasto RETURN è premuto, la linea di testa risulta introdotta nel buffer ed il tasto RUBOUT non può più essere utilizzato per la correzione di errori in quella linea.

LEADER e TRAILER - Allorchè si perfora una banda oppure si registrano delle informazioni su di un'audio cassetta, prima del testo voluto, sono posti in uscita 128 caratteri di testa (leader) (200 ottale, 80 esadecimale). Dopo che la parte di testo specificata è stata posta in uscita, sono posti in uscita anche 128 caratteri di chiusura (trailer) (200 ottale, 80 esadecimale).

TEA ritorna quindi al modo di comando per permettere l'introduzione di un altro comando. Allorchè avviene la lettura nel microcalcolatore di un nastro o di un'audio cassetta, TEA ignora sia il leader che il trailer (i caratteri del leader e del trailer non sono cioè trascritti nel buffer).

CTRL/C - Con l'introduzione di un CTRL/C, TEA ritorna immediatamente al modo di comando. Se si è introdotto un comando che, per la sua esecuzione, esige che sia premuto il tasto RETURN, un CTRL/C fa sì che il comando stesso venga ignorato. Questo naturalmente non accade nel caso che il comando non richieda un RETURN. Un CTRL/C, ad esempio, non arresterà l'esecuzione da parte di TEA di un comando H, O, M, Q, R, A, N, od X, per la semplice ragione che tali comandi sono eseguiti non appena si ha l'introduzione del comando di una sola lettera.

Nel caso che si introduca in TEA una linea di testo mentre ci si trova nel modo di testo, un CTRL/C avrà come effetto che TEA ignori la linea di testo e ritorni al modo di comando. Evidentemente, come si è già avuto modo di dire, questo avverrà soltanto se il CTRL/C è introdotto prima che il tasto RETURN sia stato premuto.

CTRL/TAB (CTRL/I) - TEA è stata scritto in modo che la telescrivente o il CRT appaiono dotati di fermi di posizione o tabulazioni (tab stop). Tali fermi sono sistemati ad ogni posizione di otto caratteri di una linea. Allorchè si preme il tasto TAB o quello I mentre è tenuto abbassato il tasto CTRL/C la testina di scrittura della telescrivente o il cursore del CRT si spostano del necessario numero di spazi verso destra per raggiungere la prossima tabulazione. A questa caratteristica si fa spesso

ricorso allorchè si introducono in TEA delle istruzioni simboliche (sorgente), da assemblare in un secondo tempo. Il ricorso a punti di fermo da luogo ad un listing organizzato in modo ordinato nel corso della terza passata dell'assembler. L'impiego di punti di fermo in luogo di spazi consente una notevole economia di memoria.

CTRL/S - La funzione del carattere CTRL/S è quella di carattere terminale di una stringa di ricerca (introdotta dopo il comando S (SEARCH). Ilprocesso di ricerca comincia dal momento in cui è introdotto il CTRL/S.

CTRL/E - Le funzioni svolte dal carattere CTRL/E sono due. Quando si introduce il comando E (EXCHANGE), bisogna introdurre un CTRL/E dopo la stringa se si vuole che lo scambio avvenga. Mentre ci si trova nel modo di testo, un CTRL/E ha come effetto che l'intera linea di testo che si sta battendo sia cancellata. TEA risponde stampando un ritorno del carrello, un avanzamento di linea e, quindi, il medesimo numero di linea relativo alla linea di testo cancellata. Non appena il tasto RETURN è premuto, la linea di testo è introdotta nel buffer ed il tasto CTRL/E non può più essere impiegato per cancellare la linea di testo.

DISPOSITIVI DI I/O

Nel Capitolo 3 sarà descritto il metodo impiegato da TEA per ottenere l'accesso ai dispositivi di I/O ed il modo di cui è possibile modificare le subroutine di I/O per adattarsi a dispositivi di I/O diversi. A questo punto è comunque opportuno esaminare i termini relativi ai dispositivi di I/O adottati nel resto del libro.

Le subroutine di I/O utilizzate da TEA per comunicare con il mondo esterno sono sei. Grazie a queste subroutine di I/O TEA può comunicare con una tastiera (IOTEST, IONOEC, IOECHO), una stampante o un CRT (IOO), un lettore di banda o un registratore a cassette in playback (funzionamento come riproduttore) (IOR) ed un perforatore di banda od un registratore a cassette nel modo di registrazione (IOP). I termini fra parentesi non costituiscono solamente i nomi delle subroutine di cui si vale TEA, ma altresì i nomi che saranno impiegati per riferirsi ai vari dispositivi di I/O. Nel caso della tastiera si impiegherà il termine IONOEC, di preferenza ai termini altrettanto corretti IOTEST e IOECHO.

Caricamento in Memoria di un Programma Simbolico con l'impiego di TEA

Il caricamento in memoria di un programma sorgente può compiersi da uno di questi due dispositivi di I/O: una tastiera ASCII oppure un dispositivo periferico nel modo di lettura (lettere di nastro, registratore a

cassette funzionante come riproduttore, ecc.). Nel caso che la sorgente provenga da una tastiera ASCII, significa che si sta utilizzando il dispositivo IONOEC.

Se si vale di un "lettore" la sorgente è invece letta da un dispositivo periferico IOR. Di norma, il programma sorgente, è all'inizio creato valendosi del dispositivo IONOEC (una tastiera ASCII). Dopo che è stato creato, il programma sorgente può essere memorizzato nel dispositivo di I/O (IOP). Tale dispositivo può essere un perforatore di banda comandato da telescrivente, un registratore a cassette o un qualsiasi altro dispositivo periferico che soddisfi le condizioni richieste al dispositivo IOP (si veda il Capitolo 3, *Convenzioni di Input/Output*). Naturalmente, il programma sorgente depositato nel dispositivo periferico può essere riletto in memoria, valendosi di TEA, in qualsiasi momento per mezzo dei comandi A (APPEND), I (INSERT) e C (CHANGE). Il programma può essere allora editato e depositato di nuovo nel dispositivo IOP, oppure assemblato valendosi dell'assembler presente all'interno di TEA. I due esempi che seguono chiariscono le modalità di impiego di un lettore di banda comandato da telescrivente e da un registratore a cassette per introdurre nel buffer un programma sorgente.

Impiego di un Lettore di Nastro Comandato da Telescrivente per Introdurre Il Programma Sorgente

Per caricare nel buffer un programma sorgente contenuto su banda, è indispensabile come prima cosa avviare TEA introducendo gli opportuni indirizzi iniziale e finale del buffer. È bene riservare al buffer quanta più memoria possibile. Se, infatti, la memoria specificata è insufficiente, può avvenire che TEA stampi il messaggio di errore ME! nel bel mezzo della banda. In questo caso l'utente deve servirsi del comando M (MEMORY) per accrescere le dimensioni del buffer e quindi ricominciare la lettura della banda in memoria.

Dal momento che TEA è stato appena avviato, in memoria non è presente alcun programma sorgente, per cui si introduce una N in risposta alla domanda "SORGENTE PRESENTE IN MEMORIA?". TEA risponde con C=, al che si preme il tasto R per abilitare il dispositivo IOR (il lettore di banda). La testa della banda è allora posta nel lettore. Premendo il tasto A (APPEND), TEA passa al modo di testo. Lo switch di selezione della telescrivente è spostato nella posizione START e la lettura in memoria della banda può compiersi.

Allorchè avviene la lettura del CTRL/C al termine della banda, TEA ritorna al modo di comando e stampa C=. Lo switch di selezione della telescrivente può essere riportato nella posizione FREE e la banda tolta. Servendosi del dispositivo IONOEC si può allora editare il buffer, oppure si può assemblare il contenuto del buffer. Si ricordi che, una volta che il CTRL/C è stato letto dalla banda, dispositivo IOR è disabilitato. Nell'esempio 1-2 è compreso più d'uno dei passi qui esaminati.

Impiego di un Registratore a Cassette per Introdurre un Programma Sorgente

La procedura da eseguire per leggere un programma sorgente da un registratore a cassette presenta molte analogie con quella appena utilizzata a proposito della banda perforata. In pratica l'unica differenza risiede nella subroutine di I/O che permette di accedere al registratore anzichè al lettore di banda.

Dopo che sono stati introdotti gli indirizzi iniziale e finale e TEA è stato informato che in memoria non è presente alcun programma sorgente, si introduce il comando R (READER). Si avvia allora il registratore nel modo PLAY (riproduci). Allorchè dal registratore avviene la lettura del leader come si può avvertire dal suono che esso produce, si; introduce il comando A (APPEND). Allorchè si ha la lettura del CTRL/C posto al termine del nastro il dispositivo IOR è disabilitato automaticamente e quello IONOEC abilitato. TEA a questo punto, può essere utilizzato per editare il programma contenuto in memoria oppure per assemblare il programma sorgente.

Allorchè si è trattato dell'impiego di un lettore di banda per leggere in memoria un programma sorgente (sezione precedente), si è detto che l'Esempio 1-2 presentava il modo in cui TEA era impiegato per questa operazione. In realtà, non vi è nessun modo di sapere se in questo esempio il programma sorgente proveniva da un lettore di banda oppure da un registratore a cassette. L'Esempio 1-2 costituisce un esempio valido sia per l'uno che per l'altro caso. Come già detto, l'unica differenza tra l'impiego del lettore di banda e quello del registratore a cassette risiede nella subroutine di I/O che assiste i differenti dispositivi di I/O. Nel Capitolo 3, *Convenzioni di Input/Output*, si avrà modo di trattare tanto le modalità secondo cui i dispositivi di I/O sono abilitati e disabilitati, quanto le necessarie istruzioni di software relative alle sei subroutine di I/O IOTEST, IONOEC, IOECHO, IOO, IOR, IOP.

CONSIDERAZIONI CONCLUSIVE SULL'EDITOR

A proposito dell'impiego dell'editor compreso in TEA è necessario ricordare alcuni punti importanti:

1. TEA provvede da sè a generare e stampare i numeri di linea mano a mano che il programma sorgente è introdotto nel buffer del testo servendosi dei comandi A (APPEND), I (INSERT), e C (CHANGE). L'utente deve preoccuparsi di introdurre i numeri solo nel caso che voglia accedere ad una particolare porzione del buffer.
2. Dopo l'introduzione di un comando A (APPEND), I (INSERT) e C (CHANGE), TEA si limita a ritornare al modo di comando nel caso che sia stato introdotto un CTRL/C.

3. TEA salta automaticamente al primo punto di fermo successivo (uno ogni otto posizioni di stampa) se si introduce un CTRL/I e un CTRL/B.
4. Se il buffer si spinge all'interno delle ultime 256 locazioni di memoria che producono l'indirizzo finale specificato dall'utente, TEA provvede a stampare il messaggio di errore ME! e quindi disabilita i comandi A (APPEND) e I (INSERT). Per cambiare una o più linee di testo, tuttavia, può essere utilizzato il comando C (CHANGE). Può, naturalmente, essere utilizzato il comando per lo scambio di stringa nel caso che il comando S (SEARCH) e quello N (NEXT) abbia scoperto una stringa corrispondente.
5. Il dispositivo IOR può essere utilizzato per inviare in ingresso al buffer del testo un programma sorgente valendosi di uno dei comandi A (APPEND), I (INSERT) e C (CHANGE).
6. Allorchè si utilizzano i comandi A (APPEND), I (INSERT) e C (CHANGE), ci si può servire del tasto RUBOUT (DELETE) per cancellare gli eventuali errori mentre il tasto line-feed può servire a stampare la linea corretta. Se si introduce un CTRL/E, l'intera linea di testo risulta cancellata. La linea di testo non è introdotta nel buffer sino a che non sia premuto il tasto RETURN.
7. L'impiego del comando E (EXCHANGE) richiede particolare attenzione. Occorre conoscere con certezza, volta per volta, quale sia la stringa scambiata e con quale stringa avvenga lo scambio. Questo comando, il più delle volte, è impiegato per correggere un certo numero di errori di ortografia uguali nei vari commenti oppure per cambiare tutti i riferimenti ad un certo indirizzo simbolico (degli indirizzi simbolici si tratterà nel prossimo capitolo).

Quale esempio conclusivo riguardante l'editor ci si riferisca all'Esempio 1-21. In esso sono utilizzati vari comandi dell'editor attraverso i quali si può osservare il modo con cui si accede al buffer, come viene modificato e posto su di un dispositivo periferico. I comandi ed il loro formato non dovrebbero essere di difficile comprensione. Tra i comandi utilizzati nell'esempio ne manca uno, il comando X (ASSEMBLE). Prima di ricorrere a questo comando è necessario esaminare quali siano i caratteri validi su cui l'assembler è in grado di operare, le varie espressioni simboliche valide, nonchè il formato dei programmi che possono essere assemblati.

Esempio 1-21. Dimostrazione dell'editor in TEA.

```
INITIAL ADDRESS ? 0050 000
FINAL ADDRESS ? 0060 000
SOURCE CURRENTLY IN MEMORY ? N
C=A
0001  /NOW IT\TS THE TIME FOR ALL GOOS\SD MEN
0002  /TO COME TO THE AID OF THIER\REIEIR
0002  /TO COME TO THE AID OF THEIR COUNTRY.
0003
0004  /TO BE OR NOT TO BE, THAT IS THE QUESTION.
0005  ?
C=L
0001  /NOW IS THE TIME FOR ALL GOOD MEN
0002  /TO COME TO THE AID OF THEIR COUNTRY.
0003
0004  /TO BE OR NOT TO BE, THAT IS THE QUESTION.
0005
C=S=MEN
0001  /NOW IS THE TIME FOR ALL GOOD MEN

C=E=WOMEN
C=L1
0001  /NOW IS THE TIME FOR ALL GOOD WOMEN

C=N?
C=I1
0001  /THIS IS A DEMONSTRATION OF THE EDITOR IN TEA.
0002  ?
C=L1-3
0001  /THIS IS A DEMONSTRATION OF THE EDITOR IN TEA.
0002  /NOW IS THE TIME FOR ALL GOOD WOMEN
0003  /TO COME TO THE AID OF THEIR COUNTRY.

C=A
0006          *010 000
0007  PRINT,  INXH     /INCREMENT THE MEMORY ADDRESS
0008          MOVAM    /GET A CHARACTER
0009          CPI      /COMPARE IT TO ZERO
0010          000
0011          RZ       /RETURN IF IT IS ZERO
0012          CALL     /OTHERWISE, PRINT THE
0013          TTYOUT   /CHARACTER
0014          0
0015          JMP      /THEN GET ANOTHER CHARACTER
0016          PRINT
0017          0
0018  ?
C=Q
051 207
060 000
LINE # =  0018

C=L1-7
0001  /THIS IS A DEMONSTRATION OF THE EDITOR IN TEA.
0002  /NOW IS THE TIME FOR ALL GOOD WOMEN
0003  /TO COME TO THE AID OF THEIR COUNTRY.
0004
0005  /TO BE OR NOT TO BE, THAT IS THE QUESTION.
0006          *010 000
0007  PRINT,  INXH     /INCREMENT THE MEMORY ADDRESS

C=D1-3

C=D2
```

Esempio 1-21. Continuazione dell'editor in TEA.

```
C=L
0001
0002            *010 000
0003  PRINT,    INXH     /INCREMENT THE MEMORY ADDRE?
C=S=PRINT
0003  PRINT,    INXH     /INCREMENT THE MEMORY ADDRESS

C=E=OUTIT
C=N
0008            CALL     /OTHERWISE, PRINT THE

C=N
0012            PRINT

C=ES
C=N?
C=E?
C=L
0001
0002            *010 000
0003  OUTIT,    INXH     /INCREMENT THE MEMORY ADDRESS
0004            MOVAM    /GET A CHARACTER
0005            CPI      /COMPARE IT TO ZERO
0006            000
0007            RZ       /RETURN IF IT IS ZERO
0008            CALL     /OTHERWISE, PRINT THE
0009            TTYOUT   /CHARACTER
0010            0
0011            JMP      /THEN GET ANOTHER CHARACTER
0012            OUTIT
0013            0
0014
C=D91-100
?
C=D9-10

C=C8
0008            CALL     /OTHERWISE, PRINT THE ASCII
0009            TTYOUT   /CHARACTER IN THE A REGISTER
0010            0        /ON THE TELETYPEWRITER OR CRT.
0011  ?

C=L
0001
0002            *010 000
0003  OUTIT,    INXH     /INCREMENT THE MEMORY ADDRESS
0004            MOVAM    /GET A CHARACTER
0005            CPI      /COMPARE IT TO ZERO
0006            000
0007            RZ       /RETURN IF IT IS ZERO
0008            CALL     /OTHERWISE, PRINT THE ASCII
0009            TTYOUT   /CHARACTER IN THE A REGISTER
0010            0        /ON THE TELETYPEWRITER OR CRT.
0011            JMP      /THEN GET ANOTHER CHARACTER
0012            OUTIT
0013            0
0014
C=M
INITIAL ADDRESS ?
FINAL ADDRESS ? 0070 000
SOURCE CURRENTLY IN MEMORY ? Y
C=P
```

Esempio 1-21. Continuazione dell'editor in TEA.

```
*010 000
OUTIT, INXH/INCREMENT THE MEMORY ADDRESS
MOVAM/GET A CHARACTER
CPI/COMPARE IT TO ZERO
000
RZ/RETURN IF IT IS ZERO
CALL/OTHERWISE, PRINT THE ASCII
TTYOUT/CHARACTER IN THE A REGISTER
0/ON THE TELETYPEWRITER OR CRT.
JMP/THEN GET ANOTHER CHARACTER
OUTIT
0

C=Q
051 032
070 000
LINE # =  0014
C=M
C=Q
29 1A
36 00
LINE # =  0014
C=D
ARE YOU SURE ? Y
C=M
INITIAL ADDRESS ?
FINAL ADDRESS ?
SOURCE CURRENTLY IN MEMORY ? N
C=R
C=A
?
C=L

0001            *042 341
0002  LOOPIT, DCRC     /DECREMENT THE COUNT
0003          JNZ      /LOOP BACK IF THE COUNT
0004          LOOPIT   /IS NON-ZERO
0005          0
0006          RET      /WHEN IT IS ZERO, RETURN
0007
C=I1
0001  /THIS IS OUR OLD FAVORITE!
0002  ?
C=L1-5
0001  /THIS IS OUR OLD FAVORITE!
0002            *042 341
0003  LOOPIT, DCRC     /DECREMENT THE COUNT
0004          JNZ      /LOOP BACK IF THE COUNT
0005          LOOPIT   /IS NON-ZERO

C=
```

CAPITOLO 2

Modalità di impiego dell'Assembler

Nel capitolo precedente si è visto come l'editor incorporato in TEA possa essere utilizzato per creare un programma sorgente. Tale programma può contenere valori relativi ai dati simboli di istruzioni, commenti e indirizzi simbolici. L'editor, in ogni caso, non ha modo di sapere che lo si sta utilizzando per editare un programma: per quanto lo riguarda, infatti, il suo impiego potrebbe avere indifferentemente lo scopo di stilare una lettera, un elenco di spedizioni o il manoscritto di un libro. L'editor di limita a manipolare caratteri e stringhe di caratteri. Esso non ha alcuna nozione, e, daltronde, non è affare che lo riguardi, circa la natura di ciò che si introduce nel buffer tramite la tastiera, il lettore di banda perforata e il registratore a cassette, ma si limita a manipolare il contenuto del buffer in conformità ai comandi dell'utente.

Purtroppo, il programma sorgente memorizzato nel buffer con l'impiego dell'editor non può essere eseguito dall'8080. Si ricordi, infatti, che le linee di testo sono composte di caratteri ASCII che l'editor non ha fatto altro che memorizzare in tante locazioni di memoria una di seguito all'altra. Le stringhe di caratteri ASCII devono essere in qualche modo convertite negli uni e zeri logici costituenti i codici operativi op-code delle istruzioni dell'8080 o i valori corrispondenti ai dati. Una volta che le linee di codice sono state convertite in qualcosa su cui l'8080 possa operare, queste devono essere caricate in memoria così che l'8080 possa prelevare dalla memoria questi valori di otto bit e trattarli, a seconda del caso, come istruzioni oppure come dati.

Nel caso che non si disponga di un editor e di un assembler, oppure che il sistema "calcolatore" sia troppo piccolo per riuscire a fungere da supporto di un editor e un assemblèr, i programmi devono essere redatti e assemblati manualmente.

Essi devono cioè, prima essere scritti su un foglio di carta e poi

impaginati, cancellando e spostando istruzioni sino a che la sequenza di istruzioni non assuma una forma appropriata. Il programma è quindi assemblato manualmente. Tale lavoro implica il prendere in considerazione i simboli nelle istruzioni contenuti nel listing del programma e cercare i corrispondenti codici operativi (op-code) su di una scheda o elenco fornito dal fabbricante. È ovvio che tale metodo di assemblaggio manuale è poco sicuro e comunque eccessivamente lento.

È proprio per aggirare questi problemi di edizione e assemblaggio dei programmi che sono stati scritti editor e assembler suscettibili di essere impiegati per modificare dei programmi sorgenti e convertirli altresì in codice macchina comprensibile per il calcolatore. In questo caso, TEA è stato scritto in modo da poter essere eseguito sui microcalcolatori basati sull'8080, 8085 o Z-80. Esso può essere impiegato per editare frasi simboliche che costituiscono altrattante istruzioni dell'8080, nonchè convertire tali istruzioni in codici operativi che l'8080 è in grado di eseguire.

CARATTERISTICHE DELL'ASSEMBLER

L'Assembler di TEA può essere utilizzato per assemblare (1) programmi completi memorizzati nel buffer del testo oppure (2) programmi memorizzati su banda perforata. Nel caso che l'assembler debba assemblare un programma leggendolo da banda, esso effettua la lettura di una sola linea del programma sorgente per volta dal dispositivo periferico. Elabora quindi la linea appena letta e, in capo ad un breve intervallo di tempo, passa a leggere dal dispositivo periferico un'altra linea del programma sorgente. Anche se, malauguratamente, la maggior parte dei registratori a cassette non possono essere utilizzati per leggere una riga del programma sorgente per volta, a questo scopo tuttavia, può servire egregiamente la maggioranza dei lettori di banda, unità floppy disk e registratori digitali.

Per assemblare un programma, indipendentemente dalla sua provenienza (memoria o banda perforata), TEA deve operare con due o tre passate; deve, cioè, esaminare dal principio alla fine il programma sorgente due o tre volte. Nel caso che il programma completo sia già presente in memoria, a TEA possono occorrere da 15 a 20 secondi per effettuare la prime due passate. Nel caso, invece, che TEA debba leggere il programma sorgente da banda perforata per le prime due passate possono essere necessari anche 10 o 20 minuti.

Nel corso della prima passata, TEA provvede a generare una tabella dei simboli degli indirizzi, assegnando cioè, ad ogni indirizzo simbolico presente nel programma un indirizzo a 16 bit. L'indirizzo così associato a ciascun indirizzo simbolico è determinato dalla sua "posizione" nel programma. Nel corso della prima passata, perciò, il lavoro svolto da TEA è poco appparisccente. Esso, infatti non comporta nè la perforazione di una banda nè la stampa di un listing del programma assemblato sul CRT o su telescrivente. Degli indirizzi simbolici e del metodo di

indirizzamento simbolico si avrà occasione di trattare più dettagliatamente nella sezione che segue in questo stesso capitolo.

Nel corso della seconda passata, TEA comincia effettivamente il processo di conversione delle stringhe di caratteri ASCII costituenti le espressioni mnemoniche (istruzioni) nei codici binari che l'8080 è in grado di eseguire. È in questa fase che TEA provvede a perforare una banda con il programma assemblato che rappresenta una *versione oggetto* del programma stesso. Tale banda perforata può essere letta nel sistema in un momento successivo valendosi del comando B (BOOTSTRAP) per far si che il programma sia eseguito. TEA inoltre, può caricare il programma assemblato *direttamente* in memoria mentre si effettua la perforazione della banda. Per eseguire un programma non è cioè necessario che la sua versione assemblata sia stata preventivamente riportata su banda perforata o registrata su audiocassetta.

TEA si avvale della terza passata per produrre un listing del programma assemblato. Nel corso di questa, TEA provvede a fare il listing del programma sorgente generato dall'utente, unitamente ai codici binari da lui stesso generati, in corrispondenza delle espressioni simboliche contenute nelle varie linee del programma sorgente e dagli indirizzi di memoria dove quei codici dovranno essere memorizzati. Il listing si dimostra davvero prezioso in tutti quei casi in cui si impiega un DEBUGGER per localizzare gli errori di un programma, fornendo informazioni complete sull'ubicazione delle subroutine, dei dati provvisori e delle istruzioni. Il listing costituisce inoltre un elemento fondamentale della indispensabile documentazione relativa ad un dato programma. Di un programma infatti, non è sufficiente possedere le istruzioni operative, ma si deve altresì conservare il listing più aggiornato, completo di note sulle modifiche e inserimenti.

Nel corso della terza passata attraverso il programma, TEA provvede anche a stampare il contenuto della tabella dei simboli degli indirizzi, la quale conterrà tutti gli indirizzi simbolici definiti dall'utente insieme agli indirizzi a 16 bit che sono stati assegnati. Questa tabella è disposta in ordine alfabetico. Al termine, nonchè alla fine di ogni passata effettuata dall'assembler attraverso il programma, è stampato il numero di errori rilevati insieme dall'assembler.

Le istruzioni che TEA deve eseguire per il corretto ordinamento della tabella dei simboli degli indirizzi sono particolarmente laboriose. Tra l'istante in cui TEA ha terminato il listing del programma e quello in cui avviene la stampa dei contenuti della tabella dei simboli degli indirizzi può dunque trascorrere un tempo non trascurabile. A titolo di esempio, per l'ordinamento di una tabella dei simboli degli indirizzi contenente 540 indirizzi simbolici, il tempo impiegato da TEA è pari a 90 se

Naturalmente, uno dei vantaggi di avere editor e assembler presenti contemporaneamente in memoria è quello di poter utilizzare l'editor per la correzione degli eventuali errori scoperti dall'assembler, ripetendo poi una seconda volta l'assemblaggio del programma. Questa caratteristica può essere sfruttata solamente nel caso che il programma da assemblare sia memorizzato tutto insieme in memoria. Se, infatti, si scoprisse un

errore mentre si sta procedendo alla lettura di una banda particolarmente lunga, sarebbe necessario rileggere nell'editor quella sezione della banda, correggere l'errore e quindi perforare un nuovo nastro. Il processo di assemblaggio dovrebbe allora riprendere dall'inizio, dal punto cioè, in cui la banda è letta in ingresso nel corso della prima passata dell'assembler. È superfluo perciò osservare come l'assemblare un programma partendo da banda perforata costituisce un compito lungo e monotono.

La funzione dell'assembler di TEA è quella di creare una versione del programma sorgente che sia eseguibile dalla macchina. Naturalmente, come si è già avuto modo di constatare, l'editor di TEA può anche essere utilizzato per evitare una qualsiasi combinazione di lettere e parole presenti nel text buffer. L'assembler, in ogni caso non può far altro che *tradurre delle specifiche sequenze di caratteri* in un *programma oggetto* che l'8080 è in grado di eseguire. Il resto di questo capitolo si occupa delle sequenze di caratteri traducibili dall'assembler, dei messaggi di errore generati in caso che queste presentino degli errori, nonchè del modo di servirsi dell'assembler per l'assemblaggio di un programma sorgente.

FRASI (STATEMENTS)

Ogni frase, ovvero linea di testo (codice), che l'assembler traduce in un'istruzione eseguibile dalla macchina o in un byte di dati si compone di un certo numero (da uno a tre) di *campi*. I campi e la loro posizione nella linea di testo (codice) sono i seguenti:

Etichetta	Codice Operativo oppure dato	/Commento dell'Utente

Il campo dell'*etichetta* (label) della frase è costituito dal nome simbolico inventato dal programmatore per identificare in un programma una particolare istruzione o un particolare dato. L'etichetta o *indirizzo simbolico* può quindi essere impiegata dal programmatore come riferimento alla locazione di memoria utilizzata per memorizzare l'istruzione o il dato. Gli indirizzi simbolici possono essere formati da un numero *qualsiasi* di lettere o da una *qualsiasi* combinazione di lettere e numeri, anche se, generalmente, la lunghezza dell'indirizzo simbolico dovrebbe essere limitata a cinque o sei caratteri. Una particolare sequenza di caratteri deve essere impiegata una sola volta nel campo delle etichette e ad ogni sequenza di caratteri disposta nel campo delle etichette deve seguire immediatamente una virgola. Allorchè nel campo delle etichette ci si serve di un indirizzo simbolico, si dice che esso è *definito*.

Il secondo campo della linea di testo (codice) può contenere o il *valore di un dato* oppure *una delle istruzioni simboliche dell'8080*. Il formato valido per questi dati e istruzioni sarà esaminato fra breve. Subito dopo

il dato o il simbolo dell'istruzione, la linea di testo conterrà (1) uno spazio, (2) un RETURN, (3) una tabulazione o ancora, (4) una barretta (per un commento).

Il campo dei commenti contiene un testo composto di parole, frasi o affermazioni aventi lo scopo di fornire chiarimenti sull'operazione eseguita dall'istruzione o sul dato contenuto nella linea di testo. Il commento può contenere qualsiasi carattere *ad eccezione del segno distintivo del dollaro ($)*. I commenti sono sempre preceduti da una barra (/).

La struttura di TEA è relativamente elastica. I commenti e le istruzioni non devono necessariamente stare in campi rigorosamente delimitati. Le parole relative alle istruzioni o ai dati non hanno alcun obbligo di trovarsi in un campo distante otto spazi dal margine di sinistra; esse possono essere disposte ovunque in una linea di testo. Se, tuttavia, una linea contiene un indirizzo simbolico, l'istruzione, il dato oppure il commento devono trovarsi alla sua destra. L'unica condizione "di struttura" che deve essere soddisfatta è che gli *indirizzi simbolici*, quando sono *definiti*, abbiano inizio al *margine di sinistra*. Nessuno spazio o tabulazione può precedere in alcun caso un indirizzo simbolico definito.

L'unico motivo per cui è consigliabile mantenere i vari elementi in campi separati (indirizzi simbolici nel campo delle etichette, dati o simboli di un'istruzione nel secondo campo e commenti nell'ultimo) consiste nel vantaggio che il programma si presenti in un formato di agevole lettura. Se gli indirizzi simbolici, i dati, i simboli relativi alle istruzioni ed i commenti fossero disposti in ordine sparso, sarebbe difficile individuare un simbolo di istruzione non corretto e far risaltare il punto in cui un indirizzo simbolico è stato effettivamente definito. Mentre l'Esempio 2-1 presenta un programma memorizzato in un formato ben ordinato, il programma il cui listing compare nell'Esempio 2-2 non è per niente chiaro. Anche se entrambi i programmi possono essere assemblati da TEA senza errori, il programma presentato dall'Esempio 2-1 darà luogo ad un listing molto più ordinato in fase di assemblaggio.

Esempio 2-1. Listing ben ordinato di un programma generato dell'editor di TEA.

```
C=L
0001          *042 341
0002  LOOPIT, DCRC     /DECREMENT THE COUNT
0003          JNZ      /LOOP BACK IF THE COUNT
0004          LOOPIT   /IS NON-ZERO
0005          0
0006          RET      /WHEN IT IS ZERO, RETURN
0007
C=
```

Esempio 2-2. Un listing disordinato dello stesso programma dell'esempio 2-1.

```
C=L
0001  *042 341
0002  LOOPIT, DCRC     /DECREMENT THE COUNT
0003  JNZ/LOOP BACK IF THE COUNT
0004          LOOPIT/IS NON-ZERO
0005  0
0006                   RET      /WHEN IT IS ZERO, RETURN
0007
C=
```

Indirizzo simbolico

Come indirizzo simbolico può essere utilizzata una generica stringa di caratteri ASCII composta di lettere, ovvero di lettere e numeri. Allorchè si definisce un indirizzo simbolico, questo deve apparire nel primo campo della linea di testo a cominciare proprio dll'inizio della linea. Esso, inoltre, non deve essere preceduto da nessuno spazio o tabulazione. All'indirizzo simbolico, così definito, è assegnato un valore di 16 bit nel "contatore dell'indirizzo attuale" (ADDCNT) allorchè TEA procede a creare la tabella dei simboli degli indirizzi nel corso della prima passata del procedimento di assemblaggio.

Esempio 2-3. Impiego di indirizzi simbolici in un programma.

```
        *010 003
TEST,   MVIA    /LOAD THE A REGISTER WITH THE ASCII
        101     /VALUE FOR THE LETTER "A"
        CALL    /THEN PRINT THE CHARACTER ON THE
        TTYOUT  /TELETYPEWRITER OR CRT.
        0
        MVIA    /LOAD THE A REGISTER WITH THE ASCII
        075     /VALUE FOR AN EQUAL SIGN, AND
        CALL    /PRINT IT.
        TTYOUT
        0
        HLT     /THEN HALT.

TTYOUT, MOVBA   /SAVE THE CHARACTER IN B
TTYO,   IN      /INPUT THE UART'S STATUS WORD.
        001
        ANI     /SAVE ONLY THE TRANSMITTER'S FLAG
        004     /IF A=004, THE PRINTER IS READY
        JZ      /IF A=000, THE PRINTER IS BUSY
        TTYO    /IF THE PRINTER IS BUSY, WAIT
        0       /FOR IT TO BE FINISHED.  THEN THE
        MOVAB   /A REGISTER CAN BE PRINTED
        OUT     /AFTER THE CHARACTER IS MOVED FROM
        000     /B TO A, OUTPUT IT TO THE UART.
        RET     /RETURN WITH THE CHARACTER IN A.
```

La scelta degli indirizzi simbolici deve tenere conto della seguente condizione limitativa: nessun indirizzo simbolico di tre caratteri può terminare con una H. Non possono, ad esempio essere impiegati come indirizzi simbolici ASH, 12H e HOH, per la ragione che essi hanno lo stesso formato standard dei numeri esadecimali, cosicchè TEA sarebbe indotto a trattarli come tali. Nel corso del procedimento di assemblaggio, TEA non stamperebbe nessun genere di messaggio di errore incontrando un indirizzo simbolico A1H o FFH.

Nell'Esempio 2-3 vi sono tre indirizzi simbolici definiti, TEST TTYOUT e TTYO. Allorchè questo programma è assemblato, a TEST è assegnato l'indirrizzo a 16 bit 010 003 (0803 esadecimale), a TTYOUT l'indirizzo 010 016 (080E esadecimale) e a TTYO l'indirizzo 010 017 (080F esadecimale). Tali indirizzi saranno utilizzati dall'assembler tutte le volte che il programma farà riferimento a TEST, TTYOUT o TTYO. Assemblando questo programma si può constatare come a questi indirizzi simbolici siano assegnati proprio questi valori di 16 bit.

Nell'Esempio 2-4 riporta il listing del programma assemblato generato nel corso della terza passata. Benchè non se ne sia ancora parlato, si

Esempio 2-4. Programma dell'Esempio 2-3 in versione assemblata.

```
ERRORS DETECTED = 000
------

      TYCHON EDITOR-ASSEMBLER V-3                          PAGE 01-001

                        *010 003
010 003 076  TEST,      MVIA     /LOAD THE A REGISTER WITH THE ASCII
010 004 101             101      /VALUE FOR THE LETTER "A"
010 005 315             CALL     /THEN PRINT THE CHARACTER ON THE
010 006 016             TTYOUT   /TELETYPEWRITER OR CRT.
010 007 010             0
010 010 076             MVIA     /LOAD THE A REGISTER WITH THE ASCII
010 011 075             075      /VALUE FOR AN EQUAL SIGN, AND
010 012 315             CALL     /PRINT IT.
010 013 016             TTYOUT
010 014 010             0
010 015 166             HLT      /THEN HALT.

010 016 107  TTYOUT,    MOVBA    /SAVE THE CHARACTER IN B
010 017 333  TTYO,      IN       /INPUT THE UART'S STATUS WORD.
010 020 001             001
010 021 346             ANI      /SAVE ONLY THE TRANSMITTER'S FLAG
010 022 004             004      /IF A=004, THE PRINTER IS READY
010 023 312             JZ       /IF A=000, THE PRINTER IS BUSY
010 024 017             TTYO     /IF THE PRINTER IS BUSY, WAIT
010 025 010             0        /FOR IT TO BE FINISHED.  THEN THE
010 026 170             MOVAB    /A REGISTER CAN BE PRINTED
010 027 323             OUT      /AFTER THE CHARACTER IS MOVED FROM
010 030 000             000      /B TO A, OUTPUT IT TO THE UART.
010 031 311             RET      /RETURN WITH THE CHARACTER IN A.
```

può osservare come l'impiego dell'asterisco (*) abbia la funzione di contrassegnare o definire un indirizzo del programma. Poichè l'asterisco è seguito dall'indirizzo 010 003, sarà questo il primo indirizzo utilizzato dal programma. Ne deriva che il TEST comincia alla locazione di memoria 010 003 (0803 esadecimale). Il codice operativo corrispondente all'istruzione simbolica MVIA (076 ottale, 3E hex) sarà memorizzato in questa locazione di memoria allorchè sarà letta in memoria la banda perforata nel corso della seconda passata dell'assembler. Il fatto notevole che non deve sfuggire in questo esempio è l'impiego dell'indirizzo 010 016 nelle due occasioni in cui l'indirizzo simbolico TTYOUT era stato richiamato dalle istruzioni CALL, nonchè l'impiego dell'indirizzo assegnato a TTYO, 010 017 (080F hex), da parte dell'istruzione JZ alla fine dell'esempio.

In molti dei prossimi esempi ci si servirà ancora di listing di programmi assemblati. Come si può constatare, agli indirizzi simbolici sono stati realmente assegnati gli indirizzi appropriati. Allorchè nell'ambito di un'istruzione ci si vale di indirizzi simbolici, come nel caso delle due istruzioni CALL e di quella JZ dell'Esempio 2-4, l'assembler fa uso degli indirizzi appropriati.

Numeri

In un programma possono essere utilizzati numeri ottali ed esadecimali, in nessun caso numeri decimali. Salvo diversa indicazione, TEA

interpreta i vari numeri come ottali. Un numero ottale si compone di alcune cifre da una a tre, comprese tra zero e sette. Per valersi di numeri esadecimali, questi devono essere di *due* cifre ed essere seguiti da un H. I numeri 2H e 01FH non sono numeri esadecimali validi.

ISTRUZIONI VALIDE E LORO FORMATO

Nel secondo campo di una linea di testo ci può servire di una delle istruzioni simboliche dell'8080 oppure di un dato di 8 bit (in forma ottale o esadecimale). In questa sezione del capitolo, si tratterà pertanto del formato da rispettare per le istruzioni a uno, due e tre byte dell'8080.

Istruzioni a Un Solo Byte

Per le istruzioni ad un solo byte l'utente non dispone di nessuna opzione. I simboli mnemonici utilizzati in molti manuali, libri e nella documentazione delle specifiche dell'8080 sono riconosciuti da TEA. Esso incorpora una tabella, detta *tabella dei simboli permanenti*, contenente tutti i simboli delle istruzioni dell'8080 accompagnati dal codice operativo a 8 bit relativo a ciascun simbolo. Le istruzioni ad un solo byte incluse nella tabella dei simboli permanenti comprendono: MOV (da registro a registro e da/a registro a/da memoria), HLT, INR, DCR, ADD, ADC, SUB, SBB, ANA, XRA, ORA, CMP, RLC, RRC, RAL, RAR, e tutte le istruzioni di ritorno (RET, RZ, RNZ, RC, RNC, RPO, RPE, RM, ed RP). La tabella dei simboli contiene anche le istruzioni di "push" e di "pop" (PUSHPW, PUSHB, PUSHD, PUSHH e POPSW, POPB, POPD e POPH) insieme alle istruzioni che riguardano copie di registri XCHG, XTHL, SPHL, PCHL, DADB, DADD, DADH, DADSP, STAXB, STAXD, LDAXB, LDAXD, INXB, INXD, INXH, INXSP, DCXB, DCXD, DCXH e DCXSP.

Nella tabella dei simboli permanenti sono altresì memorizzate le istruzioni relative al registro A e cioè CMA, STC, CMC e DAA, come pure le istruzioni di restart (RST1, RST2, RST3, RST4, RST5, RST6 ed RST7) insieme alle istruzioni EI, DI e NOP. La tabella comprende infine anche le due nuove istruzioni dell'8085 SIM e RIM. Affinchè possano essere correttamente interpretati da TEA, i programmi devono contenere questi simboli di istruzione nell'esatta forma in cui sono stati riportati. Questo significa che l'istruzione MOVAB non deve essere introdotta in un programma nella forma MOV A, B nè in quella MOVA, B. Altri simboli non validi sarebbero RST1, INX H DAD B e INR E. Non vi devono essere, cioè, spazi e virgole che separino i registri tra di loro e dal resto del simbolo. Il mancato rispetto di questo formato determina il rilevamento di almento un errore da parte di TEA nel corso sia della prima che della seconda passata dell'assemblaggio.

Il programma elencato nell'Esempio 2-5 contiene un certo numero di simboli di istruzione validi a uno e tre byte che TEA è in grado di

Esempio 2-5. Alcune istruzioni ad un solo byte valide che TEA è in grado di assemblare senza errori.

```
C=L
0001          *023 340
0002  ADDIT,  DADM     /MULTIPLY H&L BY TWO
0003          DCRB     /DECREMENT THE COUNT IN B
0004          JNZ      /IF THE RESULT IS NON-ZERO,
0005          ADDIT    /MULTIPLY H&L AGAIN
0006          0
0007          INRC     /THEN INCREMENT THE COUNT IN C
0008          DADD     /AND ADD THE VALUE IN D&E
0009          XCHG     /SWAP D&E WITH H&L
0010          RET
0011
0012  POPIT,  INXSP    /INCREMENT THE STACK POINTER
0013          DADSP    /ADD THE SP TO H&L
0014          MOVAM    /GET A DATA VALUE
0015          RRC      /ROTATE IT TO THE RIGHT
0016          SUBE     /SUBTRACT E
0017          MOVMA    /SAVE THE VALUE BACK ON THE STACK
0018          RNC      /RETURN IF THE CARRY IS ZERO
0019          INRM     /OTHERWISE INCREMENT MEMORY
0020          RET      /AND THEN RETURN
0021
```

interpretare correttamente. Il loro impiego non dà luogo a nessun messaggio di errore. L'Esempio 2-6 contiene invece alcuni simboli non validi, con il risultato che TEA stamperà altrettanti messaggi di errore dovuti, appunto, all'impiego di simboli così fatti. Gli unici simboli validi dell'esempio 2-6 sono, infatti, soltanto XCHG, RET, RNC e RRC. Tutti gli altri presentano delle virgole o degli spazi e non sono, perciò, accettabili.

Esempio 2-6. Impiego di un formato non corretto per alcune istruzioni ad un solo byte dell'8080.

```
C=L
0001          *023 340
0002  ADDIT,  DAD      H        /MULTIPLY H&L BY TWO
0003          DCR      B        /DECREMENT THE COUNT IN B
0004          JNZ      ADDIT    /IF NON-ZERO, MULTIPLY AGAIN
0005          INR      C        /INCREMENT THE COUNT IN C
0006          DAD      D        /THEN ADD D&E TO H&L
0007          XCHG              /EXCHANGE D&E WITH H&L
0008          RET
0009
0010  POPIT,  INX      SP       /INCREMENT THE STACK POINTER
0011          DAD      SP       /ADD THE SP TO H&L
0012          MOV      A,M      /GET A DATA VALUE
0013          RRC               /ROTATE IT TO THE RIGHT
0014          SUB      E        /SUBTRACT E
0015          MOV      M,A      /SAVE THE VALUE
0016          RNC               /RETURN IF THE CARRY IS ZERO
0017          INR      M        /OTHERWISE, INCREMENT MEMORY
0018          RET               /AND THEN RETURN
0019
C=
```

Istruzioni a Due Byte

Le istruzioni a due byte possono essere suddivise o in istruzioni di ingresso/uscita (I/O) oppure in istruzioni immediate. Le istruzioni di I/O possono presentare sia un numero ottale che uno esadecimale

Esempio 2-7. Formato appropriato per le istruzioni IN e OUT.

```
C=L
0001
0002            *034 156
0003  TEST,     IN       /GET A VALUE FROM THE SWITCHES
0004            034
0005            OUT      /OUTPUT IT TO THE DISPLAYS
0006            DISP
0007            RET      /RETURN WITH THE VALUE IN A
0008
0009  TEST2,    XRAA     /SET THE A REGISTER TO ZERO
0010            OUT      /OUTPUT IT TO THE D/A CONVERTER
0011            3FH
0012            OUT      /AND ALSO TO THE A/D MULTIPLEXER
0013            MUX
0014            RET      /RETURN WHEN DONE
0015
0016
C=
```

oppure anche una stringa di caratteri definita dalla pseudo-op (pseudo-operazione) DB come il codice dispositivo di un dispositivo di I/O. Il numero ottale, o quello esadecimale, o la stringa di caratteri, costituisce il secondo byte dell'istruzione a due byte, mentre il primo byte sarà il simbolo dell'istruzione IN o OUT. L'Esempio 2-7 include un certo numero di istruzioni di I/O valide nel formato ad esse appropriato.

Nell'Esempio 2-7 la subroutine TEST dapprima pone in ingresso un dato proveniente dalla porta di ingresso 034 (ottale) (uno C hex), e quindi pone in uscita lo stesso dato alla porta di uscita alla quale è stata associata la stringa di caratteri DISP. Mediante una delle pseudo-op di TEA, è possibile assegnare un numero di 8 bit (ottale ovvero hex) ad una stringa di caratteri, come, appunto, DISP. In TEST2 si azzera il registro A mediante l'istruzione XRAA e quindi se ne pone in uscita il contenuto alla porta di uscita 3F (077 ottale) ed a quella MUX. L'8080 ritorna a questo punto dalla subroutine TEST2.

La subroutine, il cui listing è riportato nell'Esempio 2-8, contiene cinque simboli non validi. I secondi byte delle istruzioni IN e OUT non devono essere contenuti nella stessa linea di testo in quanto gli indirizzi delle porte devono essere contenuti in quella immediatamente successiva, come illustrato nell'Esempio 2-7. Inoltre l'istruzione XRAA non deve presentare uno spazio o un punto di fermo tra XRA ed A.

Le istruzioni immediate ammettono o un numero ottale, ovvero un numero esadecimale o una stringa di caratteri definita dalla pseudo-op

Esempio 2-8. Formato non corretto per le istruzioni IN e OUT dal quale consegue che TEA genera alcuni messaggi di errore.

```
C=L
0001            *034 156
0002  TEST,     IN       034      /GET A VALUE FROM THE SWITCHES
0003            OUT      DISP     /OUTPUT IT TO THE DISPLAYS
0004            RET               /RETURN WITH THE VALUE IN A.
0005
0006  TEST2,    XRA      A        /LOAD THE A REGISTER WITH ZERO
0007            OUT      3FH      /OUTPUT IT TO THE D/A CONVERTER
0008            OUT      MUX      /ALSO OUTPUT IT TO THE A/D MULTIPLEXER
0009            RET               /RETURN WHEN DONE
0010
C=
```

Esempio 2-9. Impiego di numeri ottali, esadecimali, caratteri tra virgolette e byte definiti (stringhe di caratteri) quali byte di dato per alcune istruzioni e due byte dell'8080.

```
C=L1-18
0001            *010 000
0002  KEYIN,    IN      /GET A CHARACTER FROM THE
0003            TTY     /ASCII KEYBOARD
0004            CPI     /IS THE A KEY PRESSED?
0005            "A"
0006            JNZ     /NO, THEN WAIT FOR IT
0007            KEYIN   /TO BE PRESSED.
0008            0
0009            MVIA    /THE A KEY IS PRESSED, SO
0010            34H     /OUTPUT THE VALUE 34 (OCTAL 64)
0011            OUT     /TO SOME SEVEN-SEGMENT DISPLAYS
0012            SSD
0013            ADI     /ADD AN IMMEDIATE DATA BYTE
0014            105     /OF 105 (HEX 45) AND OUTPUT
0015            OUT     /THE RESULT TO SOME DISCRETE LEDS
0016            203
0017            HLT     /THEN HALT.
0018
```

DB o, infine, un carattere tra virgolette. Le istruzioni immediate in oggetto comprendono MVIA, MVIB, MVIC, MVID, MVIE, MVIH, MVIL, MVIM, ADI, ACI, SUI, SBI, ANI, XRI, ORI, e CPI. Le istruzioni IN e OUT, insieme a quelle appena viste, sono le uniche istruzioni a due byte riconosciute da TEA. L'Esempio 2-9 contiene alcune istruzioni di questo tipo con la dimostrazione delle differenti modalità in cui queste possono essere utilizzate. Nell'Esempio 2-9 l'8080 pone in ingresso un dato dal dispositivo di I/O definito mediante le stringa TTY. Tale dato è caricato nel registro A dell'8080 dove avviene il confronto con l'equivalente ASCII di A. Allorchè TEA assembla la linea di testo che contiene il carattere tra virgolette, secondo byte dell'istruzione immediata, utilizza come byte di dato immediato l'equivalente ASCII del carattere tra virgolette. Nel corso della seconda passata dell'assembler, l'equivalente ASCII sarà perforata su banda o registrato su cassetta. Esso apparirà anche nella stampa del listing generato con la terza passata dell'assembler.

Se il tasto A dell'Esempio 2-9 non è premuto, l'8080 salta indietro a KEYIN. Se, invece, il tasto A è premuto, il registro A è caricato, per effetto dell'istruzione MVIA, con il numero esadecimale 34 (064 ottale). Tale valore è quindi posto in uscita verso alcuni display a sette segmenti definiti dalla stringa di caratteri SSD. Al contenuto del registro A è poi sommato un byte immediato pari a 105 (45 hex) ed il risultato di questa somma è posto in uscita verso il dispositivo di uscita 203 (83 hex), dopo di che l'8080 si ferma.

Allorchè provvede ad assemblare il programma dell'Esempio 2-9, TEA stampa un indirizzo ed un valore relativo ai dati per ogni linea di testo. Questo fa sì che, allorchè passa a stampare la linea che contiene TTY/ASCII KEYBOARD, TEA debba stampare il valore definito *in precedenza* come equivalente alla stringa TTY. L'8080 dovrà, inoltre, procedere in modo del tutto analogo a proposito della stampa del secondo byte, SSD, dell'istruzione OUT. Dovrà cioè, cercare il valore da stampare in una tabella dove esso è definito come equivalente alla

stringa SSD. Al momento di assemblare l'istruzione CPI, TEA determinerà l'*equivalente ASCII del carattere tra virgolette*. In questo caso l'equivalente ASCII della lettere A è 101 (41 hex), valore che sarà appunto stampato insieme all'indirizzo di memoria in cui sarà memorizzato in occasione del caricamento del programma in memoria.

L'Esempio 2-10 contiene due istruzioni a due byte scritte in un formato non corretto. Il secondo byte dell'istruzione MVIA deve trovarsi nella linea immediatamente successiva a quella del simbolo MVIA. Tra i due byte dell'istruzione non possono esservi, infatti, ne linee vuote e neppure linee contenenti commenti. Analogamente, l'indirizzo di dispositivo dell'istruzione OUT non deve essere separato dal simbolo OUT da uno spazio vuoto o da un commento. Se si prova ad assemblare il programma dell'Esempio 2-10, la precedenza di questi due errori fa stampare due messaggi di errore. Come si può notare, *il secondo byte dell'istruzione a due byte deve trovarsi nella prima linea di tasto che segue, immediatamente dopo quella contenente il simbolo mnemonico dell'istruzione*. Poichè, nell'esempio 2-10, tale regola non è rispettata, ne derivano dei messaggi di errore.

Esempio 2-10. Metodo non corretto di impiego di istruzioni a due byte.

```
C=L1-9
0001            *010 000
0002  START,    MVIA     /LOAD THE A REGISTER WITH
0003
0004            123      /123 (HEX 53)
0005            OUT      /OUTPUT THE VALUE TO SOME
0006                     /SEVEN-SEGMENT DISPLAYS THAT
0007            003      /ARE INTERFACED TO THE 8080
0008            HLT      /THEN HALT
0009
```

Istruzioni a Tre Byte

L'8080 è dotato di un certo numero di istruzioni a tre byte, in cui ciascun byte dell'istruzione deve essere memorizzato in una distinta linea di testo. Le istruzioni a tre byte necessitano, perciò, di tre linee di testo consecutive, proprio come per la memorizzazione delle istruzioni a due byte occorrono due linee di testo consecutive e per quelle ad un solo byte una singola linea di testo. I byte secondo e terzo delle istruzioni a tre byte possono consistere in due numeri ottali, due numeri esadecimali, un numero ottale ed uno esadecimale, un indirizzo simbolico definito internamente al programma o un indirizzo simbolico che è stato definito mediante una pseudo-operazione. Un indirizzo simbolico definito all'interno del programma consiste in un indirizzo simbolico che si trova nel primo campo di una linea di testo.

Questo comporta che in quella linea davanti ad esso non figurano ne spazi ne tabulazioni e che nella linea stessa, immediatamente dopo l'indirizzo simbolico, sia memorizzata una virgola. Indirizzi simbolici così definiti sono già stati visti ed utilizzati in precedenza.

Se, come secondo e terzo byte di un'istruzione a tre byte, si impiegano due numeri ottali, o esadecimali, oppure un numero ottale ed uno

esadecimale, ciascuno numero *deve* essere posto in una distinta linea di testo. Se nell'istruzione a tre byte si impiega un indirizzo simbolico, indipendentemente da come lo si sia definito, la stringa di caratteri che lo rappresenta deve essere posta nella seconda linea. Per seconda linea si intende la linea di testo immediatamente seguente quella che contiene il simbolo dell'istruzione. La terza linea dell'istruzione deve contenere uno zero, in quanto questo zero nella terza linea dell'istruzione ha la funzione di "lasciare lo spazio" all'assembler per sistemare il byte alto dell'indirizzo dell'istruzione nel corso della seconda (perforazione di una banda) e terza passata (listing del programma assemblato) del processo di assembling.

Le istruzioni a tre byte che possono valersi delle varie combinazioni di numeri ottali ed esadecimali e degli indirizzi simbolici comprendono tutte le istruzioni di salto (JMP, JNZ, JNC, JZ, JC, JPO, JPE, JP e JM); tutte le istruzioni di chiamata (CALL, CNZ, CZ, CNC, CC, CPO, CPE, CP e CM); e le istruzioni di caricamento e memorizzazione a 16 bit (LXIB, LXID, LXIH, LXISP, LHLD ed SHLD). Pure comprese nel gruppo delle istruzioni a tre byte sono le istruzioni STA ed LDA operanti sul contenuto del registro A. In particolare si osservi come nelle istruzioni LXIB, LXID, LXIH ed LXISP non siano presenti spazi vuoti ne tabulazioni e neppure virgole. Se a queste istruzioni fossero aggiunti degli spazi, tabulazioni o delle virgole, TEA non sarebbe in grado di assemblare correttamente i simboli e ne risulterebbero dei messaggi di errore.

Nel listing dell'Esempio 2-11 sono presenti alcune istruzioni a tre byte. Dal momento che la prima istruzione di chiamata (CALL) utilizza un indirizzo simbolico, la terza linea dell'istruzione contiene uno zero. Allorchè TEA procede ad assemblare tale istruzione, esso stampa gli otto bit low dell'indirizzo di 16 bit assegnato a TTYIN insieme alla linea

Esempio 2-11. Impiego di numeri ottali, esadecimali ed indirizzi simbolici per il secondo e terzo byte delle istruzioni a tre byte.

```
C=L1-23
0001            *AFH 00H
0002  BYTE3,    CALL     /GET A CHARACTER FROM THE
0003            TTYIN    /TELETYPEWRITER'S KEYBOARD.
0004            0
0005            CPI      /IS THE A KEY PRESSED?
0006            "A"
0007            CZ       /YES, THEN PRINT A CARRIAGE
0008            172      /RETURN AND LINE FEED ON THE
0009            1CH      /TELETYPEWRITER
0010            JMP      /THEN JUMP BACK TO BYTE3
0011            BYTE3
0012            0
0013
0014  LOAD,     LHLD     /LOAD H&L WITH THE CONTENT OF
0015            TEMPO    /MEMORY AT "TEMPO" AND "TEMPO+1"
0016            0
0017            SHLD     /THEN SAVE THE 16-BIT VALUE
0018            CHARIN   /IN "CHARIN" AND "CHARIN+1"
0019            0
0020            JMP      /THEN JUMP TO THE SYSTEM MONITOR
0021            000      /IN EPROM.
0022            374
0023
```

di testo "TTYIN/TELETYPEWRITER'S KEYBOARD". Allorchè TEA stampa la terza linea di testo, gli otto bit alti dell'indirizzo di 16 bit sono stampati insieme allo zero contenuto in tale linea di testo. L'istruzione a tre byte seguente, CZ, presenta come secondo byte un numero ottale ed un numero esadecimale come terzo byte. Ne consegue che, premendo il tasto A della tastiera della telescrivente, l'8080 effettua una chiamata della subroutine memorizzata in memoria all'indirizzo 1CH 172 (034 172 ottale, 1C7A hex). Come ormai dev'essere noto, allorchè la terza istruzione a tre byte, JMP, è assemblata, l'8080 provvede a leggere l'indirizzo di 16 bit corrispondente a BYTE3 nella tabella dei simboli degli indirizzi che TEA ha composto nel corso della prima passata, indirizzo che sarà stampato in occasione della stampa della seconda e terza linea dell'istruzione JMP.

La seconda parte dell'Esempio 2-11 contiene tra le istruzioni a tre byte. TEA provvederà a cercare nella tabella degli indirizzi gli indirizzi di 16 bit corrispondenti a TEMPO e CHARIN, che saranno poi utilizzati nel corso della seconda e terza passata dell'assembler. I due numeri ottali impiegati nell'istruzione JMP saranno assemblati come secondo e terzo byte dell'istruzione, poichè TEA non ha bisogno di far riferimento alla tabella dei simboli degli indirizzi per determinare l'appropriato indirizzo di 16 bit in base ai numeri ottali 000 e 374.

Al pari delle istruzioni a due byte, anche quelle a tre byte devono essere comprese tutte in tre linee di testo *consecutive*.

Nell'Esempio 2-12, tutte e tre le istruzioni a tre byte sono scritte in un formato non corretto. Il secondo e terzo byte della prima istruzione CALL sono infatti separati dalla prima linea dell'istruzione dalla presenza di un commento. Lo zero della terza linea di testo che segue CALL TO LPRINT è separato dagli altri due byte dell'istruzione da una linea vuota. Nell'ultima istruzione a tre byte il simbolo JMP è separato dall'indirizzo simbolico START e dallo zero da una linea vuota. Se si volesse assemblare il programma dell'Esempio 2-12, si otterebbero dei messaggi di errore. Come si può osservare, *il secondo e terzo byte dell'istruzione devono essere memorizzati nelle prime due linee di testo immediatamente seguenti alla linea di testo che contiene il simbolo mnemonico dell'istruzione stessa.*

L'Esempio 2-12 contravviene appunto a questa regola.

Esempio 2-12. Istruzioni a tre byte utilizzate in modo non corretto.

```
C=L1-14
0001            *023 145
0002   START.    CALL     /GET A CHARACTER FROM THE
0003                      /TELETYPEWRITER'S KEYBOARD
0004            TTYIN    /AND RETURN WITH IT IN THE
0005            0        /A REGISTER
0006            CALL     /THEN PRINT THE CHARACTER
0007            LPRINT   /ON THE LINE PRINTER AND
0008
0009            0        /RETURN WITH THE CHARACTER IN A.
0010            JMP      /THEN GET ANOTHER CHARACTER
0011
0012            START    /FROM THE KEYBOARD AND
0013            0        /PRINT IT.
0014
```

PSEUDO-OP (pseudo-operazioni)

Anche se già si è avuto occasione di osservare il funzionamento di una pseudo-operazione (pseudo-op), può essere che tale circostanza, per qualcuno, sia passata inosservata. Il carattere asterisco (*) costituisce una pseudo-op in quanto essa informa l'assembler incorporato in TEA che i caratteri che vengono dopo di esso (numeri ottali o esadecimali) *non* devono essere interpretati come un'istruzione dell'8080, bensì devono, invece, essere trattati come un indirizzo. Le pseudo-op, dunque, determinano l'esecuzione di particolari operazioni da parte dell'assembler. TEA è dotato di quattro pseudo-op le cui caratteristiche di funzionamento sono presentate qui di seguito.

Asterisco (*)

In molti degli esempi fin'ora incontrati si è avuto modo di osservare la presenza di un asterisco all'inizio di un programma o di una subroutine. La funzione dell'asterisco è quella di segnalare all'assembler che nel programma è stato appena definito un nuovo indirizzo. Mediante l'impiego dell'asterisco, è l'*utente* che decide *personalmente* il punto esatto in memoria in cui qualsiasi istruzione o dato sarà memorizzato in fase di assemblaggio del programma e di caricamento in memoria della versione oggetto di quel programma. Indipendentemente dal numero di volte che l'asterisco è utilizzato per la definizione di un nuovo indirizzo di 16 bit, tutti gli indirri devono essere definiti usando lo stesso formato.

Un indirizzo definito mediante l'asterisco deve seguire immediatamente quest'ultimo. Esso può essere indifferentemente composto di due numeri ottali, due numeri esadecimale oppure una combinazione qualsiasi dei due. Tra il primo byte di 8 bit dell'indirizzo ed il secondo byte di 8 bit dello stesso indirizzo deve comunque esservi uno, e soltanto uno, spazio vuoto. Anche se un programma può contenere un qualsiasi numero di indirizzi definiti mediante l'asterisco, questi devono tuttavia conformarsi tutti a questo formato. La mancata definizione di *almeno un indirizzo del programma* dà luogo ad un messaggio di errore. La funzione dell'asterisco presenta strette analogie con quella della pseudo-op ORG presente in altri assembler.

Contrassegno del Dollaro ($)

La funzione del contrassegno del dollaro ($) è quella di segnalare all'assembler incorporato in TEA che è arrivato al termine del programma sorgente nel corso del suo procedimento di assemblaggio. Qualunque simbolo mnemonico di istruzione, indirizzo simbolico, dato o commento memorizzato nel buffer del testo dopo il segno del dollaro non è preso in considerazione. Per questo motivo, *un segno del dollaro non deve mai essere presente in un commento nè essere utilizzato come byte di dato immediato tra virgolette.* Come già noto, TEA può servire ad

assemblare un programma memorizzato internamente in memoria come pure un programma memorizzato su banda perforata e suddiviso in blocchi di 100-200 linee di testo.

Nel caso che TEA stia provvedendo ad assemblare un programma memorizzato interamente in memoria, non è necessario che esso termini con un segno del dollaro. L'assembler contenuto in TEA conosce, infatti, il punto *esatto* in memoria in cui il programma sorgente termina. Se, tuttavia, il programma sorgente è assemblato da banda perforata, al termine dell'ultimo blocco del programma bisogna che sia perforato sulla banda un contrassegno di dollaro. Questo si ottiene, naturalmente, memorizzando un segno di dollaro nel text buffer valendosi dell'editor perforando, quindi, il contenuto del buffer sulla banda di uscita. L'assenza di un segno di dollaro al termine dell'ultimo blocco della banda perforata *non* darà luogo ad un messaggio di errore, bensì avrà per effetto che TEA si comporti come se di seguito vi fosse un altro blocco di testo. TEA proseguirà, perciò, leggendo un altro blocco di testo dal dispositivo IOR di I/O. Ne risulta che, se si dimentica il segno del dollaro, non sarà possibile superare la fase di prima passata dell'assembler. Anche se, per i programmi che sono assemblati dalla memoria, un segno di dollaro alla fine non è strettamente necessario, è tuttavia buona norma memorizzare comunque un segno di dollaro al termine del programma. Se, infatti, il programma cresce al punto di non poter più essere memorizzato integralmente in memoria, il segno del dollaro sarà introdotto automaticamente sulla banda perforata all'atto del trasferimento del buffer.

Il segno del dollaro che conclude il programma non compare stampato in fondo al programma stesso in occasione del listing delle terza passata. Esso è equivalente alla pseudo-op END di altri assembler.

Define Byte (definisci il byte)

Questa pseudo-op (defined byte: DB) ha la funzione di assegnare un certo numero ottale o esadecimale (otto bit) ad una stringa di caratteri. Tale valore, assegnato dall'utente alla stringa, sarà impiegato dall'assembler nel corso della seconda e terza passata. Ogniqualvolta l'assembler incontra nel programma quella stessa stringa, esso perforerà su banda il valore assegnato dall'utente (seconda passata), oppure ne elencherà il valore (terza passata). Il corretto formato da osservare per la pseudo-op DB è mostrato nell'Esempio 2-13.

In questo esempio tutte le stringhe di caratteri sono separate da un "DB" da uno spazio, ed uno solo. Il numero ottale o esadecimale è pure esso separato allo stesso modo dalla stringa di caratteri. La pseudo-op DB deve essere utilizzata unicamente *all'inizio* di un programma, prima di qualunque indirizzo simbolico, indirizzo definito (*) o simbolo di istruzione. Anche se essa può trovarsi in uno qualsiasi dei tre campi del formato dell'assembler, è consigliabile, se non altro per chiarezza, che essa cominci all'inizio del secondo campo.

Esempio 2-13. Impiego della pseudo-op DB per assegnare un valore ottale o esadecimale di otto bit ad una stringa di caratteri.

```
C=L
0001            DB START 001
0002            DB SSD 3FH
0003            DB CR 015
0004            DB LF 012
0005
0006            *076 354
0007   CRLF,    MVIA     /LOAD THE A REGISTER WITH THE
0008            CR       /ASCII VALUE FOR A CARRIAGE RETURN
0009            CALL     /PRINT THE VALUE ON THE
0010            TTYOUT   /TELETYPEWRITER OR CRT.
0011            0
0012            MVIA     /LOAD THE A REGISTER WITH THE
0013            LF       /ASCII VALUE FOR A LINE FEED
0014            CALL     /PRINT THE VALUE ON THE
0015            TTYOUT   /TELETYPEWRITER OR CRT
0016            0
0017            MVIA     /THEN LOAD THE A REGISTER WITH
0018            START    /A VALUE FOR THE SEVEN-SEGMENT
0019            OUT      /DISPLAYS AND OUTPUT IT
0020            SSD
0021            HLT      /THEN HALT
0022   $
0023
C=
```

I byte definiti possono essere utilizzati unicamente come byte di dato immediato (per le istruzioni matematiche, logiche e di spostamento di dati di tipo immediato e come byte di indirizzo di dispositivo, per le istruzioni a due byte IN e OUT). È chiaro che utilizzare un byte definito in un'istruzione ad un solo byte non *riveste un grande interesse*. Si può anche pensare di utilizzare due byte definiti per il secondo e terzo byte di un'istruzione a tre byte, ma a questo scopo si dispone di un'altra pseudo-op apposita (assegnazione di un numero di 16 bit come valore di una stringa di caratteri).

Nell'Esempio 2-13 sono comprese alcune pseudo-op. Tutti i byte definiti sono stati definiti in base alla pseudo-op DB prima di qualsiasi simbolo mnemonico di istruzione o indirizzo definiti (*076 354). Se dei byte di 8 bit fossero definiti con la pseudo-op DB in qualsiasi altro punto del programma che non sia la "testa" del programma stesso, si avrebbe un messaggio di errore.

Provvedendo ad assemblare il programma dell'Esempio 2-13, TEA assegnerà dei valori di 8 bit alle quattro stringhe ASCII.

Il byte di dato immediato della prima istruzione NVIA riceverebbe il valore 015 (0D hex) ed il byte di dato immediato della seconda istruzione MVIA avrebbe il valore 012 (0A hex). Assemblando la terza istruzione MVIA, si impiegherà il byte di dato immediato 001 (01 hex) è, per l'istruzione OUT, l'indirizzo di dispositivo della porta di uscita sarà 3F (077 ottale). Si osservi la presenza del segno di dollaro al termine di questo programma.

Define Word (definisci la parola)

La pseudo-op define-word (definisci la parola: DW) serve ad asse-

gnare un numero ottale o esadecimale di 16 bit ad una stringa di caratteri. Il valore che l'utente assegna alla stringa sarà impiegato dall'assembler nel corso della seconda e terza passata. Ogniqualvolta incontra quella stessa stringa, l'assembler perfora o lista il valore assegnato dall'utente. Le parole definite possono essere utilizzate per formare delle liste di valori, oppure impiegate in istruzioni a tre byte in modo del tutto analogo agli indirizzi simbolici (la seconda linea contiene le stringa, la terza uno zero). Il formato corretto per la pseudo-op di definizione di parola (DW) è illustrato nell'Esempio 2-14.

In questo esempio la stringhe di caratteri SYSMON, DEBUG, STACK e TTYIN sono separate da "DW" con uno, ed uno solo, spazio vuoto così come si usa un solo spazio per separare la stringa dagli otto bit più significativi del numero di 16 bit. Anche gli otto bit meno significativi del numero sono separati da un solo spazio da quelli più significativi. Si osservi come, analogamente alle stringhe definite dalla pseudo-op DB, le stringhe definite dalla pseudo-op DW devono apparire proprio all'inizio del programma precedendo qualunque simbolo mnemonico di istruzione, indirizzo simbolico o indirizzo definito (*).

Nell'assemblare il programma elencato nell'Esempio 2-14, TEA si servirà degli opportuni numeri di 16 bit ogniqualvolta incontrerà la stringa definita dalla pseudo-op DW nel secondo e terzo byte di un'istruzione TEA utilizzerà, dunque, l'indirizzo 170 80H (170 200 ottale, 7880 hex) come secondo e terzo byte dell'istruzione LXISP e l'indirizzo FDH 043 (375 043 ottale, FD23 hex) come secondo e terzo byte dell'istruzione CALL. Come si può immaginare, anche gli indirizzi definiti corrispondenti a SYSMON e DEBUG saranno impiegati nei punti opportuni.

TEA stamperà un messaggio di errore tutte le volte che come secondo byte di un'istruzione a due o tre byte si impiega una stringa che non sia

Esempio 2-14. Impiego della pseudo-op DW per assegnare un indirizzo di 16 bit ad una stringa di caratteri.

```
C=L1-24
0001            DW SYSMON 374 000
0002            DW DEBUG 80H 00H
0003            DW STACK 170 80H
0004            DW TTYIN FDH 043
0005
0006            *000 000
0007   START,    LXISP    /LOAD THE STACK POINTER WITH
0008            STACK    /A R/W MEMORY ADDRESS.
0009            0
0010            CALL     /GET A CHARACTER FROM THE
0011            TTYIN    /TELETYPEWRITER OR CRT.
0012            0
0013            CPI      /WAS AN "S" ENTERED?
0014            "S"
0015            JZ       /YES, THEN JUMP TO THE
0016            SYSMON   /SYSTEM MONITOR PROGRAM.
0017            0
0018            CPI      /WAS A "D" ENTERED?
0019            "D"
0020            JZ       /YES, THEN JUMP TO THE
0021            DEBUG    /ASSEMBLY LANGUAGE
0022            0        /DEBUGGER PROGRAM
0023            HLT      /NOT AN S OR A D, SO HALT
0024   $
```

Esempio 2-15. Impiego della pseudo-op DW per assegnare un numero di 16 bit ad una stringa di caratteri.

```
C=L1-14
0001            DW COUNT 001 370
0002
0003            *003 341
0004  WAIT,     LXID     /LOAD REGISTER PAIR D WITH
0005            COUNT    /A 16-BIT NUMBER TO BE
0006            0        /DECREMENTED
0007  WAIT1,    DCXD     /DECREMENT THE COUNT
0008            MOVAD    /GET THE MSBY OF THE COUNT
0009            ORAE     /OR IT WITH THE LSBY
0010            JNZ      /IF THE COUNT IS NON-
0011            WAIT1    /ZERO, JUMP BACK TO THE
0012            0        /DCXD INSTRUCTION
0013            RET      /THEN RETURN WHEN THE COUNT IS ZERO
0014  $
```

stata definita nè mediante una pseudo-op DB o DW, nè come indirizzo simbolico definito all'interno del programma).

Il motivo che spiega questo comportamento, ovviamente, è che TEA non ha modo di determinare il valore (i valori) da impiegare come secondo o secondo e terzo byte dell'istruzione.

La pseudo-op DW non ha solo la funzione di definire un'indirizzo di una subroutine o l'indirizzo di partenza di un altro programma, come nell'Esempio 2-14 in cui essa è stata, per l'appunto, utilizzata per definire indirizzi di 16 bit. Se lo si desidera, la pseudo-op DW può anche servire a definire un numero da utilizzare per un conteggio o che sia utile per altre operazioni di manipolazione dei dati, come appunto appare nell'Esempio 2-15.

In questo esempio si assegna alla stringa COUNT il valore 001 370 (01F8). Provvedendo ad assemblare il simbolo di istruzione LXID, TEA si accorge che si tratta di un'istruzione a tre byte, cosicchè il secondo e terzo byte possono essere numeri ottali o esadecimali, un indirizzo simbolico od una stringa definita dalla pseudo-op DW. Ne risulta che, come secondo byte dell'istruzione, si utilizza il valore 370 (F8 hex) e quello 001 (01 hex) come terzo byte. Si ricordi come in tutte le istruzioni a tre byte gli otto bit meno significativi del numero di 16 bit sono impiegati come secondo byte mentre gli otto bit più significativi costituiscono il terzo byte. L'esempio 2-16 contiene la versione assemblata della subroutine elencata nell'Esempio 2-15.

Esempio 2-16. Assemblaggio della subroutine listata nell'esempio 2-15.

```
                        DW COUNT 001 370

                        *003 341
003 341 021  WAIT,      LXID     /LOAD REGISTER PAIR D WITH
003 342 370             COUNT    /A 16-BIT NUMBER TO BE
003 343 001             0        /DECREMENTED
003 344 033  WAIT1,     DCXD     /DECREMENT THE COUNT
003 345 172             MOVAD    /GET THE MSBY OF THE COUNT
003 346 263             ORAE     /OR IT WITH THE LSBY
003 347 302             JNZ      /IF THE COUNT IS NON-
003 350 344             WAIT1    /ZERO, JUMP BACK TO THE
003 351 003             0        /DCXD INSTRUCTION
003 352 311             RET      /THEN RETURN WHEN THE COUNT IS ZERO
```

CARATTERI SPECIALI

In molti esempi precedenti si è già potuto osservare l'effetto di alcuni *caratteri speciali* caratterizzati da un'azione analoga a quella della pseudo-op. Essi, cioè, fanno sì che l'assembler di TEA esegua delle operazioni particolari invece di interpretare la stringa di caratteri come un valore numerico, un simbolo di istruzione o un indirizzo simbolico. I caratteri che si sono già incontrati sono l'asterisco (*; impiegato per la definizione di indirizzi), il contrassegno del dollaro ($; impiegato per denotare la fine di un programma) ed il carattere virgolette ("; impiegato allorchè occorre un valore ASCII come byte di dato di un'istruzione immediata). Tra i caratteri che TEA considera come speciali ve ne sono ancora altri.

Più (+) e Meno (—)

Questi due caratteri servono al programmatore per sommare (+) oppure sottrarre (—) uno scostamento dall'indirizzo assegnato ad un indirizzo simbolico. Lo scarto può essere espresso da un qualsiasi numero ottale o esadecimale compreso fra 000 377, ossia tra 00 ed FF. L'Esempio 2-17 illustra il modo di utilizzare un byte di scostamento.

Esempio 2-17. Somma di un byte di scostamento ad un indirizzo simbolico.

```
C=L1-14
0001            DW COUNT 001 370
0002
0003            *003 341
0004  WAIT,     LXID     /LOAD REGISTER PAIR D WITH
0005            COUNT    /A 16-BIT NUMBER TO BE
0006            0        /DECREMENTED
0007            DCXD     /DECREMENT THE COUNT
0008            MOVAD    /GET THE MSBY OF THE COUNT
0009            ORAE     /OR IT WITH THE LSBY
0010            JNZ      /IF THE COUNT IS NON-
0011            WAIT+3   /ZERO, JUMP BACK TO THE
0012            0        /DCXD INSTRUCTION
0013            RET      /THEN RETURN WHEN THE COUNT IS ZERO
0014  $
```

In questo esempio compare la stessa subroutine listata in origine nell'Esempio 2-15, con una modifica. Anzichè impiegare gli indirizzi simbolici WAIT e WAIT1, tutti i riferimenti a WAIT1 sono stati tramutati in WAIT più un certo scostamento, invece di saltare a WAIT1, l'8080 salterà all'indirizzo uguale alla somma di WAIT più tre (ottale). Come si può ben immaginare, il risultato finale non cambia; l'8080 effettua ancora un salto indietro all'istruzione DCXD. Dalla versione assemblata si questa subroutine elencata nell'Esempio 2-18, si può osservare come l'8080 abbia, in pratica, effettuato un calcolo consistente nel sommare tre all'indirizzo di 16 bit assegnato a WAIT, utilizzandone poi il risultato come secondo e terzo byte dell'istruzione JNZ. Da un confronto dei listing dell'Esempio 2-16 e di quello 2-18, si può constatare come i programmi assemblati che ne risultano siano completamente uguali.

Esempio 2-18. Assemblaggio di una subroutine contenente uno scostamento sommato ad un indirizzo simbolico.

```
                        DW COUNT 001 370

                        *003 341
003 341 021  WAIT,      LXID     /LOAD REGISTER PAIR D WITH
003 342 370             COUNT    /A 16-BIT NUMBER TO BE
003 343 001             0        /DECREMENTED
003 344 033             DCXD     /DECREMENT THE COUNT
003 345 172             MOVAD    /GET THE MSBY OF THE COUNT
003 346 263             ORAE     /OR IT WITH THE LSBY
003 347 302             JNZ      /IF THE COUNT IS NON-
003 350 344             WAIT+3   /ZERO, JUMP BACK TO THE
003 351 003             0        /DCXD INSTRUCTION
003 352 311             RET      /THEN RETURN WHEN THE COUNT IS ZERO
```

Un rischio insito nell'impiego di uno scostamento (sia esso ottale o esadecimale) dipende dal fatto che il programmatore deve determinare il numero da sommare o sottrarre affinchè TEA generi l'indirizzo desiderato. Benchè nell'Esempio 2-17 questo sia relativamente agevole, si danno dei casi in cui la determinazione dello scostamento ottale o esadecimale corretto comporta un lavoro ben più arduo, specialmente quando lo scostamento è maggiore di 30 o 40. Un'altro problema che spesso si accompagna all'impiego dello scostamento consiste nel fatto che, in seguito all'aggiunta o all'eliminazione di istruzioni, bisogna provvedere a cambiare anche lo scostamento, come mostrano gli Esempi 2-19 e 2-20.

Nell'Esempio 2-19 sono state aggiunte alla subroutine alcune istruzioni di push (PUSHD e PUSHPSW) e pop (POPPSW e POPD). Lo scopo per cui queste istruzioni sono state aggiunte alla sobroutine è di far sí che la soubrotine non "influenzi" nessuno dei registri. Dato che all'inizio della subroutine sono state aggiunte due istruzioni, WAIT + 3 avrebbe dovuto essere cambiato in WAIT + 5, ma il programmatore se ne è dimenticato. Ne risulta che, allorchè la subroutine è assemblata (Esempio 2-20), TEA calcola un indirizzo sbagliato per il secondo e terzo byte

Esempio 2-19. Aggiunta di istruzioni ad una subroutine che utilizza uno scostamento con un indirizzo simbolico.

```
C=L
0001            DW COUNT 001 370
0002
0003            *003 341
0004  WAIT,     PUSHD    /SAVE LOE ON THE STACK
0005            PUSHPSW  /AND THEN THE PSW
0006            LXID     /LOAD REGISTER PAIR D WITH
0007            COUNT    /A 16-BIT NUMBER TO BE
0008            0        /DECREMENTED
0009            DCXD     /DECREMENT THE COUNT
0010            MOVAD    /GET THE MSBY OF THE COUNT
0011            ORAE     /OR IT WITH THE LSBY
0012            JNZ      /IF THE COUNT IS NON-
0013            WAIT+3   /ZERO, JUMP BACK TO THE
0014            0        /DCXD INSTRUCTION
0015            POPPSW   /THE COUNT IS ZERO, POP THE
0016            POPD     /PSW AND REGISTER PAIR D
0017            RET      /THEN RETURN WHEN THE COUNT IS ZERO
0018  $
0019
C=
```

Esempio 2-20. Assemblaggio della subroutine contenente istruzioni addizionali ed uno scostamento (Esempio 2-19).

```
                        DW COUNT 001 370

                        *003 341
003 341 325   WAIT,     PUSHD    /SAVE D&E ON THE STACK
003 342 365             PUSHPSW /AND THEN THE PSW
003 343 021             LXID     /LOAD REGISTER PAIR D WITH
003 344 370             COUNT    /A 16-BIT NUMBER TO BE
003 345 001             0        /DECREMENTED
003 346 033             DCXD     /DECREMENT THE COUNT
003 347 172             MOVAD    /GET THE MSBY OF THE COUNT
003 350 263             ORAE     /OR IT WITH THE LSBY
003 351 302             JNZ      /IF THE COUNT IS NON-
003 352 344             WAIT+3   /ZERO, JUMP BACK TO THE
003 353 003             0        /DCXD INSTRUCTION
003 354 361             POPPSW   /THE COUNT IS ZERO, POP THE
003 355 321             POPD     /PSW AND REGISTER PAIR D
003 356 311             RET      /THEN RETURN WHEN THE COUNT IS ZERO
```

dell'istruzione JNZ. Se si cercasse di eseguire la subroutine listata nell'Esempio 2-20, l'8080 effettuerebbe un salto alla locazione di memoria 003 344 nel caso che il contenuto della coppia di registri DE, dopo il decremento, presentasse un valore diverso da zero. Anche se in questa locazione di memoria non è memorizzata alcuna istruzione, salterebbe indietro lo stesso ed eseguirebbe uno dei byte del dato dell'istruzione LXID come se si trattasse di un'istruzione! È ovvio che questo caso non deve assolutamente accadere.

Come si può constatare dagli Esempi 2-19 e 2-20, gli scostamenti devono essere utilizzati *con la massima attenzione*. Si consiglia, nella pratica, di limitarne l'impiego più che possibile. Servendosi di essi, è spesso inevitabile commettere degli errori, mentre può risultare complicato ricordarsi di tutti i punti in cui vi sono scostamenti da cambiare via via che a (da) un programma o a (da) una sobroutine si aggiungano (si eliminano) delle istruzioni. Si osservi come, impiegando esclusivamente degli indirizzi simbolici anzichè indirizzi simbolici con scostamento positivo o negativo, il compito di determinare il corretto indirizzo per le istruzioni a tre byte sia affidato tutto all'assembler. Si ricordi che il motivo principale per cui si ricorre ad un assembler è proprio quello di sollevare il programmatore dai compiti più monotoni, dove è facile sbagliare.

Come si è già accennato, lo scostamento può anche essere sottratto all'indirizzo simbolico. A questo scopo, invece del carattere più (+), si usa quello meno (—). L'Esempio 2-21 contiene il listing di una subroutine che si vale di uno scostamento da sottrarre ad un indirizzo simbolico e l'esempio 2-22 riporta la versione assemblata della stessa subroutine.

Nell'Esempio 2-21 il programmatore ha stabilito che, nel caso che il carattere letto da una tabella sia minore del carattere ASCII A, l'8080 debba effettuare un salto indietro all'istruzione MVIA. Invece di valersi di un altro indirizzo simbolico, si è deciso che sarebbe stato altrettanto comodo sottrarre due dall'indirizzo di 16 bit assegnato all'indirizzo simbolico LOOP. Allorchè si provvede ad assemblare questa subroutine (Esempio 2-22), l'indirizzo al quale è memorizzata l'istruzione MVIA

Esempio 2-21. Sottrazione di uno scostamento da un indirizzo simbolico.

```
C=L
0001            DW SRCH 037 146
0002
0003            *007 375
0004   FINDIT,  MVIB     /LOAD THE B REGISTER WITH
0005            003      /THE TERMINATION CHARACTER
0006            MVIA     /LOAD THE A REGISTER WITH THE
0007            060      /TABLE TERMINATOR
0008   LOOP,    CALL     /EXAMINE AN ENTRY IN THE TABLE
0009            SRCH
0010            0
0011            CPI      /FIND AN ASCII A ?
0012            "A"
0013            JC       /NO, LESS THAN A, KEEP LOOKING
0014            LOOP-2
0015            0
0016            JNZ      /NO, EQUAL TO A, KEEP LOOKING
0017            LOOP
0018            0
0019            HLT      /FOUND IT, HALT.
0020
```

risulta essere di due locazioni rispetto all'indirizzo di memoria dove è memorizzata l'istruzione CALL. Nell'Esempio 2-22 è possibile constatare come TEA abbia generato il corretto indirizzo di memoria per l'istruzione JC, e, in particolare, come dall'indirizzo di 16 bit di LOOP sia sottratto lo scostamento (di 8 bit). Quest'ultima circostanza è facile da verificare in quanto all'indirizzo simbolico LOOP è assegnato l'indirizzo 010 001 e, sottraendo due da esso si ottiene come risultato 007 377, utilizzato appunto come secondo e terzo byte dell'istruzione JC.

Indipendentemente dal fatto di aggiungere un numero ad un indirizzo simbolico oppure di sottrarlo da esso, occorre andar molto cauti nell'impiego degli scostamenti. Questo riveste particolare importanza dopo che un programma è stato editato in quanto può accadere che gli scostamenti debbano essere cambiati. Si ricordi che lo scostamento può essere compreso tra 000 e 377 ossia tra 00 ed FF. Benchè, a prima vista, questa possa apparire come una limitazione, vi è da considerare che è assai improbabile che un programmatore si trovi ad aver bisogno di uno scostamento maggiore di 377 o FF, data l'estrema difficoltà di un calcolo

Esempio 2-22. Assemblaggio di un programma contenente uno scostamento che è sottratto da un indirizzo simbolico.

```
                        DW SRCH 037 146
                        *007 375
007 375 006   FINDIT,   MVIB     /LOAD THE B REGISTER WITH
007 376 003             003      /THE TERMINATION CHARACTER
007 377 076             MVIA     /LOAD THE A REGISTER WITH THE
010 000 060             060      /TABLE TERMINATOR
010 001 315   LOOP,     CALL     /EXAMINE AN ENTRY IN THE TABLE
010 002 146             SRCH
010 003 037             0
010 004 376             CPI      /FIND AN ASCII A ?
010 005 101             "A"
010 006 332             JC       /NO, LESS THAN A, KEEP LOOKING
010 007 377             LOOP-2
010 010 007             0
010 011 302             JNZ      /NO, EQUAL TO A, KEEP LOOKING
010 012 001             LOOP
010 013 010             0
010 014 166             HLT      /FOUND IT, HALT.
```

del genere. Per gli scostamenti si possono usare sia numeri ottali che esadecimali; in questo secondo caso è obbligatorio memorizzare una H terminale.

Nell'Esempio 2-21, perciò, si sarebbe potuto utilizzare uno 02 esadecimale, ma la linea 14 avrebbe dovuto essere tramutata in LOOP-02H.

Virgolette

Come si è già detto nel corso di questo capitolo, TEA può generare il valore ASCII corrispondente ad un carattere tra virgolette, nel caso che questo sia il byte di dato immediato di un'istruzione immediata matematica, logica o di spostamento. I caratteri tra virgolette non possono, tuttavia, essere impiegati come secondo byte nelle istruzioni di I/O (IN e OUT). L'editor incorporato in TEA ignora tutti i caratteri di controllo diversi da CTRL/C, CTRL/S, CTRL/E, CTRL/I o CTRL/TAB. È dunque impossibile includere tra virgolette un carattere di controllo allo scopo di utilizzarlo come byte di dato immediato.

Avendo necessità di utilizzare come tale il valore corrispondente ad un carattere di controllo, si può ricorrere alla pseudo-op DB unitamente ad una conveniente stringa di caratteri ed al codice ASCII corrispondente a quel carattere di controllo. Se, ad esempio, occorre procedere al confronto con CTRL/F di tutti i caratteri provenienti da una tastiera di telescrivente o di un CRT, si può aggiungere la linea di testo DB CTRL 006 all'inizio del programma. La stringa CTRLF può in tal modo essere utilizzata come byte di dato immediato per un'istruzione CPI.

Barra (/)

Al principio di ogni commento deve essere riportata una barra (/). Della presenza di tale barra all'inizio di ogni commento si sono già incontrati svariati esempi. Al posto della (/) molti assembler si servono del punto e virgola (;).

Virgola (,)

Al termine di ogni indirizzo simbolico definito nel campo delle etichette bisogna far uso di una virgola. Anche a questo proposito dell'impiego della virgola si sono già incontrati molti esempi relativi ad indirizzi simbolici definiti. Si noti che, allorchè un indirizzo simbolico o una stringa di caratteri sono definiti mediante la pseudo-op DW, la virgola non deve mai essere usata.

ASSEMBLAGGIO DEI PROGRAMMI

Come si è già detto nel Capitolo 1, con l'introduzione del comando X (ASSEMBLE) si ottiene che TEA cominci l'esecuzione dell'assembler. La prima informazione fornita dall'utente nel corso di questo procedi-

mento riguarda il tipo di numeri, ottale o esadecimale, che sarà impiegato nella stampa del listing. Dato che i programmi da assemblare possono essere memorizzati interamente in memoria oppure essere derivati da una banda perforata, TEA pone la domanda "M OR T?". A seconda di dove si trovano i programmi da assemblare, l'utente risponderà con M (il programma da assemblare è memorizzato in memoria) oppure con T (il programma da assemblare una linea alla volta è memorizzato su banda perforata). TEA porrà a questo punto un'altra domanda, "AUTO-LOAD?", prima di iniziare il procedimento di assemblaggio. La risposta a questa domanda può essere tanto N (NO) quanto Y (YES), a seconda che l'utente desideri o meno che TEA *memorizzi in memoria il programma assemblato mano a mano che l'assembler effettua la seconda passata attraverso il programma sorgente.* Nel caso che la risposta sia N, il programma assemblato (versione oggetto) non sarà memorizzato in memoria nel corso del procedimento di assemblaggio, ma sarà semplicemente posto in uscita verso il dispositivo IOP (perforatore di banda, registratore a cassette, ecc.). Nel caso che la risposta sia Y (YES), il programma oggetto sarà memorizzato in memoria nel corso della seconda passata dell'assembler e, contemporaneamente, sarà anche posto in uscita verso il dispositivo IOP.

L'Esempio 2-23 dà un'idea di come si svolga questo 'dialogo' iniziale con TEA. Dopo avere introdotto nel buffer del testo un breve programma, si introduce il comando X (ASSEMBLE). In risposta a ciò TEA stampa OCTAL, comunicando in tal modo che il listing della terza passata avverrà con numeri ottali. TEA chiede a questo punto dove è memorizzato il programma sorgente da assemblare e l'utente risponde con M poichè il programma è interamente contenuto in memoria. La risposta all'ultima domanda "AUTO-LOAD?", stabilirà se, nel corso della seconda passata dell'assembler, la versione oggetto del programma dovrà essere memorizzata o meno in memoria. Indipendentemente dalla risposta, *la versione oggetto del programma sarà comunque posta in uscita verso il dispositivo IOP (perforatore di banda, registratore a cassette, ecc.).* Poichè la risposta dell'utente è N, la versione oggetto del programma non sarà memorizzata in memoria nel corso del procedimento di assemblaggio.

Una volta fornita la risposta alla seconda domanda, TEA comincia il procedimento di assemblaggio. Nel corso della prima passata, TEA "costruisce" la tabella dei simboli degli indirizzi e, dal momento che durante questa operazione non si sono verificati errori, risponde con "ERRORS DETECTED = 000" (errori rilevati = 0) al termine della prima passata. Si procede allora ad effettuare la seconda passata e si pone in uscita verso il dispositivo IOP la versione oggetto del programma sorgente, senza che, contemporaneamente, essa sia memorizzata in memoria.

Per la seconda volta, non avendo rilevato errori nel corso del procedimento di assemblaggio, TEA stampa "ERRORS DETECTED = 000".

Nel corso della terza passata conclusiva attraverso il programma

Esempio 2-23. Assemblaggio di un programma.

```
C=A
0001            *040 000
0002  START,    JMP      /THIS IS THE ASSEMBLER
0003            START    /DEMONSTRATION PROGRAM.
0004            0
0005  ?
C=X
OCTAL
M OR T ? M
AUTO-LOAD ?N
ERRORS DETECTED = 000
* *
ERRORS DETECTED = 000
------

      TYCHON EDITOR-ASSEMBLER V-3                          PAGE 01-001

                        *040 000
040 000 303  START,     JMP      /THIS IS THE ASSEMBLER
040 001 000             START    /DEMONSTRATION PROGRAM.
040 002 040             0

------

      TYCHON EDITOR-ASSEMBLER V-3                          PAGE 01-002

START  =040 000
ERRORS DETECTED = 000


------
```

sorgente memorizzato in memoria, TEA provvede a stampare un listing del programma assemblato. Quindi stampa sei lineette (------), seguite dal nome dell'editor-assembler, il numero della versione e il numero della pagina. Poichè trattasi della prima pagina del listing e il programma assemblato si trovava in memoria, TEA stampa un numero di blocco pari a 01, seguito dal numero di pagina 001. Segue l'elenco del programma sorgente insieme al necessario corredo di indirizzi ed informazioni relative ai dati. Giungendo al termine del programma sorgente, TEA realizza tanti ritorni di carrello e avanzamenti di linea quanti occorrono al completamento della pagina. Sono quindi stampate una seconda volta le lineette, il nome ed il numero della versione, unitamente al numero della pagina immediatamente consecutiva, 01-002. Poichè l'assemblaggio dell'intero programma è terminato, TEA stampa il contenuto *ordinato* della tabella dei simboli degli indirizzi. Poichè in questo programma è stato impiegato un unico indirizzo simbolico, la tabella dei

simboli ne contiene solamente uno, START, al quale TEA ha assegnato l'indirizzo 040 000 nel corso della prima passata. Una volta stampati tutti gli indirizzi simbolici, con i valori assegnati a ciascuno di essi, TEA stampa il numero degli errori che sono stati rilevati nel corso della terza passata. Come si può constatare, non si è incontrato nessun errore. Di nuovo, TEA provvede a realizzare un numero di ritorni di carrello e di avanzamenti di linea sufficienti a completare la pagina, tornando poi al modo di comando.

Il ritorno al modo di comando fa sì che la correzione degli errori con l'impiego dell'editor sia particolarmente agevole nel caso che l'assemblaggio del programma avvenga da memoria (M). Nel caso invece che gli errori si siano manifestati mentre il programma sorgente era assemblato da banda (T), anzichè da memoria (M), l'impiego dell'editor è pur sempre di aiuto nella loro correzione anche se la procedura risulta così un pò più complicata.

In qualunque momento, mentre TEA è occupato a stampare le informazioni del caso sul dispositivo 100 (la stampante della telescrivente o lo schermo del CRT), si può introdurre un CTRL/C per porre fine al procedimento di assemblaggio facendo sì che TEA ritorni al modo di comando. Se ne deduce che, nel corso del procedimento di assemblaggio interviene un qualche errore, per procedere alla sua correzione non è necessario attendere il completamento del listing della terza passata.

In un'altra sezione di questo capitolo si avrà ancora l'occasione per trattare nei particolari il procedimento di assemblaggio. Ciò che importa, per il momento, è di rendersi conto che, all'inizio di questo procedimento TEA formula delle domande e che le risposte a queste domande hanno un effetto ben preciso su quanto avviene in seguito. Ora che si è esaminato il procedimento di assemblaggio, si possono passare in rassegna gli errori che TEA è in grado di rilevare in fase di assemblaggio nonchè le modalità con cui TEA li segnala.

ERRORI DI ASSEMBLAGGIO

Nel corso del processo di assemblaggio, TEA comunica all'utente tutti i casi in cui esso non riesce a interpretare una particolare sequenza di caratteri. Per esempio, i simboli di istruzione MOVABC, LXIE, PUSHC, JUMP, CALLIT ed RE non sono validi in quanto nessuno di essi rappresenta una qualche istruzione eseguibile dall'8080 o dall'8085. Se, dunque, TEA s'imbatte nel corso del programma in uno di questi simboli o in qualunque altro simbolo di istruzione non valido, esso deve darne notizia, anzi, non deve limitarsi a riferire che si sia verificato un errore, ma deve anche far sapere di che tipo di errore si è trattato e dove esso si è verificato. Se, infatti, TEA riferisse all'utente che, in fase di manipolazione di un programma consistente in 1100 linee di testo, sono stati trovati tre errori, questi si troverebbe in grave difficoltà, e perderebbe comunque molto tempo, per localizzare tali errori. Nel caso, invece, che TEA riferisca che gli errori si trovano alla linee 345, 678, e

705, sarà assai facilitato per il programmatore il compito di localizzare gli errori e, come è augurabile, di correggerli.

Purtroppo l'unica specie di errori che TEA può rilevare è quella degli *errori di sintassi*. È questo il nome degli errori che si commettono allorchè le linee di testo non corrispondono al formato descritto al principio di questo capitolo o quando si utilizzano dei simboli di istruzione non validi, come quelli del paragrafo precedente. se un indirizzo simbolico è definito due volte, se si usa una sola virgoletta in un byte di dato immediato oppure se si utilizza in un programma un numero come 19E, in tutti questi casi TEA provvede a stampare un messaggio di errore. Un tipo differente di errore che TEA *non è in grado* di rilevare è un errore di natura logica. se, ad esempio, il programmatore utilizza in una subroutine i registri sbagliati, carica e preleva dallo stack i registri nell'ordine sbagliato o punta con lo stack pointer ad un indirizzo di memoria inesistente, TEA non genera nessun messaggio di errore.

Le specie di errori previsti da TEA sono due: errori *fatali* ed errori *non-fatali*. TEA può rilevare tre tipi di errori fatali e cinque di errori non-fatali. La scoperta di un errore fatale fa si che TEA provveda a stampare un messaggio di errore, ritornando subito dopo al modo di comando, senza insistere nell'esecuzione dell'assemblaggio del programma. Questo dipende dal fatto che l'errore è tanto grave da impedire a TEA di proseguire nel procedimento di assemblaggio senza commettere a sua volta degli errori. Nel caso invece, che TEA rilevi un errore non-fatale, si ha la generazione di un messaggio di errore, senza che, per questo, TEA interrompa il procedimento di assemblaggio. Indipendentemente dalla specie di errore che ha avuto luogo, se mai ve ne è stato uno, non si deve assolutamente cercare di eseguire la versione oggetto (la versione assemblata) del programma. Questo programma oggetto può essere memorizzato su di un dispostivo periferico (perforatore di banda o registratore a cassette) oppure può essere già stato caricato automaticamente in memoria nel corso della seconda passata dell'assembler.

Messaggi di Errore

TEA provvede a stampare messaggi di errore nel corso sia della prima, sia della seconda, sia della terza passata del procedimento di assemblaggio, in tutti i casi in cui non sia in grado di interpretare una linea di testo o che esista una qualche ambiguità. Allorchè ha luogo un errore, TEA provvede a stampare un messaggio di errore sul dispositivo IOO (la stampante della telescrivente o lo schermo del CRT) insieme al numero del blocco ed al numero della linea dove l'errore si è presentato. Assemblando un programma proveniente dalla memoria, il numero di blocco del messaggio di errore è invariabilmente 01. Assemblando invece il programma da banda perforata, il numero di blocco nei messaggi di errore è compreso tra 01 e 99, a seconda del numero dei blocchi già passati in rassegna dall'assembler al momento in cui ci si è imbattuti nell'errore.

Utilizzando come dispositivo di I/O per TEA una telescrivente con un lettore/perforatore di banda, il messaggio di errore risulta stampato sulla stampante. Comunque, nel corso della seconda passata, la versione assemblata del programma è pure perforata su banda.

Ne consegue che, se nel corso della seconda passata si incontrano degli errori, i messaggi di errore sono stampati sulla stampante nonchè perforati sulla banda. Il modulo oggetto così perforato, perciò, è del tutto inservibile in quanto proprio in mezzo ai valori dei dati sono perforati dei messaggi di errore.

Esempio 2-24. Doppia definizione di un indirizzo simbolico all'interno del programma.

```
C=L
0001            *000 000
0002   START,    CALL     /WAIT FOR THE C REGISTER
0003            LOOPIT   /TO BE DECREMENTED TO ZERO
0004            0
0005            HLT      /THEN HALT
0006
0007   LOOPIT,  DCRC     /DECREMENT THE COUNT
0008            JNZ      /LOOP BACK IF THE COUNT
0009            LOOPIT   /IS NON-ZERO
0010            0
0011            RET      /WHEN IT IS ZERO, RETURN
0012
0013   LOOPIT,  XRAA     /SET THE A REGISTER TO ZERO
0014            RET
0015
C=X
OCTAL
M OR T ? M
AUTO-LOAD ?N
REDEF  AT LINE # 01-0013

ERRORS DETECTED = 001

ERRORS DETECTED = 000
------


      TYCHON EDITOR-ASSEMBLER V-3                          PAGE 01-001


                        *000 000
000 000 315   START,    CALL     /WAIT FOR THE C REGISTER
000 001 004             LOOPIT   /TO BE DECREMENTED TO ZERO
000 002 000             0
000 003 166             HLT      /THEN HALT

000 004 015   LOOPIT,   DCRC     /DECREMENT THE COUNT
000 005 302             JNZ      /LOOP BACK IF THE COUNT
000 006 004             LOOPIT   /IS NON-ZERO
000 007 000             0
000 010 311             RET      /WHEN IT IS ZERO, RETURN

000 011 257   LOOPIT,   XRAA     /SET THE A REGISTER TO ZERO
000 012 311             RET
------


      TYCHON EDITOR-ASSEMBLER V-3                          PAGE 01-002


LOOPIT =000 004  START  =000 000
ERRORS DETECTED = 000
```

Doppia Definizione di un Indirizzo Simbolico (REDEF)

Benchè non sia stato esplicitamente detto, una stringa di caratteri può essere utilizzata una volta sola per definire un indirizzo simbolico. Nel caso che la stessa stringa di caratteri sia utilizzata due o più volte, TEA provvede a stampare un messaggio di errore "REDEF AT LINE # xx-nnnn", dove xx rappresenta il numero del blocco ed nnnn quello della linea di testo contenente la stringa di caratteri che è stata definita una volta di troppo.

Nell'Esempio 2-24 si può constatare come il programma contenga due indirizzi simbolici definiti in modo identico. Dopo la risposta alle due domande "M OR T?" e "AUTO-LOAD?", TEA inizia la prima passata, nel corso della quale costruisce la tabella dei simboli degli indirizzi e stampa il messaggio di errore REDEF AT LINE # 01-0013. TEA provvede quindi a completare la prima passata e stampa "ERRORS DETECTED = 001". Si effettua quindi la seconda passata. Come si può constatare, questo tipo di definizione ripetuta non può essere rilevata nè nel corso della seconda nè della terza passata, per cui TEA perfora su banda la versione oggetto del programma e poi informa l'utente che nel corso della seconda passata non è stato rilevato alcun errore.

Il listing del programma assemblato che segue avviene mentre TEA completa la terza passata attraverso il programma sorgente. Anche qui, le definizioni ripetute degli indirizzi simbolici non possono essere rilevate, per cui TEA provvede a listare l'intero programma assemblato stampando al termine il contenuto della tabella dei simboli degli indirizzi alla pagina 01-002. Anche nel corso di questa passata non è rilevato nessun errore per definizione ripetuta.

Si osservi come, nonostante gli indirizzi simbolici definiti con LOOPIT siano due, TEA si valga dell'indirizzo corrispondente alla prima presenza della stringa anche nel resto del programma. Allorchè, nel corso della seconda passata, avviene la perforazione della banda, dopo i codici operativi delle istruzioni CALL e JNZ risulta perforato su banda l'indirizzo 000 004. Sempre l'indirizzo 000 004 risulta elencato nella tabella degli indirizzi simbolici (pagina 01-002) dopo la stringa LOOPIT. Ciò che conta è che, definendo un indirizzo simbolico due o più volte nel "corpo" dello stesso programma, l'indirizzo poi utilizzato nella banda perforata, nei listing del programma assemblato e nella tabella dei simboli degli indirizzi sarà quello relativo alla prima comparsa.

Dopo aver stampato il contenuto della tabella dei simboli degli indirizzi al termine della terza passata, TEA fa ritorno al modo di comando. Dal momento che il programma che è stato assemblato è memorizzato interamente in memoria, l'editor può a questo punto essere utilizzato per eliminare o cambiare la seconda stringa LOOPIT. Il programma può quindi essere assemblato di nuovo con il risultato di ottenere un programma oggetto esente da errori.

Nell'Esempio 2-25 si verifica un errore REDEF dovuto al fatto che la stessa stringa di caratteri è stata utilizzata nel programma due volte: una volta con la pseudo-op DW (definisci la parola) ed una seconda volta

Esempio 2-25. Definizione ripetuta di un indirizzo simbolico con la pseudo-op DW.

```
C=L
0001            DW LOOPIT 123 345
0002
0003            *042 341
0004  LOOPIT,   DCRC     /DECREMENT THE COUNT
0005            JNZ      /LOOP BACK IF THE COUNT
0006            LOOPIT   /IS NON-ZERO
0007            0
0008            RET      /WHEN IT IS ZERO, RETURN
0009
C=X
OCTAL
M OR T ? M
AUTO-LOAD ?N
ERRORS DETECTED = 000

REDEF  AT LINE # 01-0001

ERRORS DETECTED = 001
------
      TYCHON EDITOR-ASSEMBLER V-3                          PAGE 01-001


                        DW LOOPIT 123 345

                        *042 341
042 341 015  LOOPIT,    DCRC     /DECREMENT THE COUNT
042 342 302             JNZ      /LOOP BACK IF THE COUNT
042 343 341             LOOPIT   /IS NON-ZERO
042 344 042             0
042 345 311             RET      /WHEN IT IS ZERO, RETURN

------
      TYCHON EDITOR-ASSEMBLER V-3                          PAGE 01-002

LOOPIT =042 341
ERRORS DETECTED = 000
```

come indirizzo simbolico definito. Come si può constatare questo ripetersi della definizione sfugge nel corso della prima passata dell'assembler. Il motivo di ciò è che qualunque stringa definita dalla pseudo-op DW è memorizzata nella tabella dei simboli degli indirizzi, unitamente al valore di 16 bit ad essa assegnata, *nel corso della seconda passata*. Di conseguenza questo tipo di definizione ripetuta può essere rilevata solo in occasione della seconda passata dell'assembler.

Allorchè ha termine l'esecuzione della prima passata, LOOPIT è memorizzato nella tabella dei simboli degli indirizzi insieme all'indirizzo 042 341. Nel corso della seconda passata, la stringa loopit associata alla pseudo-op DW è confrontata con tutte le stringhe già memorizzate nella tabella dei simboli degli indirizzi per rilevare la presenza di eventuali definizioni ripetute. Dal momento che essa è già contenuta nella tabella, si ha la generazione di un messaggio di errore. È questa la ragione per cui il messaggio di errore risulta essere REDEF AT LINE # 01-001 anzichè REDEF AT LINE # 01-004.

Come nei precedenti esempi di definizione ripetuta, il listing relativo alla terza passata non contiene nè errori nè messaggi di errore. Si osservi come l'indirizzo a 16 bit che compone il secondo e terzo byte dell'istruzione JNZ non sia altro che l'indirizzo simbolico di LOOPIT definito nel

programma e non piuttosto, l'indirizzo assegnato in base alla pseudo-op DW. Questo stesso indirizzo si trova poi stampato alla pagina 01-002 in occasione della stampa del contenuto della tabella dei simboli degli indirizzi.

Esempio 2-26. Numeri non validi che danno origine a messaggi di errore BN.

```
C=L
0001            DW TTYIN 134 005
0002
0003            *AFH00H
0004  BYTE3,    CALL     /GET A CHARACTER FROM THE
0005            TTYIN    /TELETYPEWRITER'S KEYBOARD
0006            0
0007            CPI      /IS THE A KEY PRESSED?
0008            009
0009            MVIA     /LOAD THE A REGISTER WITH
0010            2H       /2 (HEXADECIMAL)
0011            HLT
0012  $
0013
C=D13
?
C=X
OCTAL
M OR T ? M
AUTO-LOAD ?N
BN AT LINE # 01-0003

ERRORS DETECTED = 001

BN AT LINE # 01-0003

BN AT LINE # 01-0008

BN AT LINE # 01-0010

ERRORS DETECTED = 003
------

      TYCHON EDITOR-ASSEMBLER V-3                          PAGE 01-001

                        DW TTYIN 134 005

BN AT LINE # 01-0003
                        *AFH00H
000 000 315  BYTE3,     CALL     /GET A CHARACTER FROM THE
000 001 005             TTYIN    /TELETYPEWRITER'S KEYBOARD
000 002 134             0
000 003 376             CPI      /IS THE A KEY PRESSED?

BN AT LINE # 01-0008
000 004 000             009
000 005 076             MVIA     /LOAD THE A REGISTER WITH

BN AT LINE # 01-0010
000 006 000             2H       /2 (HEXADECIMAL)
000 007 166             HLT

------

      TYCHON EDITOR-ASSEMBLER V-3                          PAGE 01-002

BYTE3  =000 000  TTYIN  =134 005
ERRORS DETECTED = 003
```

Numeri Errati o Non Validi (Bad Numers: BN)

Un altro tipo di messaggio di errore è stampato allorchè TEA si attende un numero ottale oppure esadecimale e non trova nè l'uno nè l'altro. I numeri 097, 0A e 008, per esempio, non sono numeri ottali validi così come non sono numeri esadecimali validi i numeri 377, G9 e HH. L'esempio 2-26 include alcuni numeri non validi che TEA incontra nel corso della prima, seconda e terza passata dell'assembler.

Nel corso della prima passata, mentre procede alla compilazione della tabella dei simboli degli indirizzi, TEA scopre che l'indirizzo *AFH00H non è valido. È chiaro, come già si è detto, che tra il primo ed il secondo byte esadecimale dovrebbe esservi uno spazio libero. Dal momento che TEA non controlla, nel corso di questa prima passata, la validità dei simboli delle istruzioni, è questo l'unico errore scoperto. Allorchè procede ad eseguire la seconda passata attraverso il programma sorgente, TEA scopre la non validità dello stesso indirizzo, insieme a quella del byte di dato immediato dell'istruzione CPI, poichè 009 non è un numero ottale valido. TEA conclude inoltre che il byte di dato immediato dell'istruzione MVIA, 2H, non è un numero valido nè ottale nè esadecimale.

L'unica differenza tra la seconda e la terza passata consiste nel fatto che TEA, nel corso della seconda passata, perfora sui banda un programma oggetto mentre; nel corso della terza, provvede a stampare il listing del programma assemblato. Il *processo* di assemblaggio non cambia minimamente. Ne consegue che, nel corso della terza passata, sono rivelati gli stessi tre errori. Si osservi anche come, al termine di ciascuna passata, sia stampato il messaggio "ERRORS DETECTED". Anche se sono stati rilevati, in totale, "sette" errori, il numero degli errori contenuto nel programma è di tre soltanto.

Mancata Inclusione dello Zero (NZ)

Come già si è avuto accasione di ricordare nelle sezioni precedenti di questo capitolo, il secondo e terzo byte di un'istruzione a tre byte possono consistere sia in numeri ottali o esadecimali, sia in un indirizzo simbolico, sia in una stringa definita mediante la pseudo-op DW. Ricorrendo ad un indirizzo simbolico o ad una stringa definita mediante la pseudo-op DW, la linea di testo immediatamente successiva al simbolo di istruzione deve contenere la stringa di caratteri, mentre la terza linea di testo deve contenere uno zero. La presenza dello zero è necessaria per "lasciare spazio" a TEA che deve inserire gli otto bit più significativi del codice dell'indirizzo o del dato.

Nell'Esempio 2-27, dopo l'indirizzo simbolico dell'istruzione JNZ è stato dimenticato lo zero. In fase di assemblaggio del programma riportato nell'Esempio 2-27, la scoperta di tale errore avviene sia nel corso della seconda che della terza passata. Si vede immediatamente in quale punto del listing del programma assemblato avrebbe dovuto essere posto lo zero. Il formato corretto dell'istruzione JNZ è, chiaramente,

```
JNZ
LOOPIT
0
```

Esempio 2-27. Mancanza di uno zero dopo l'indirizzo simbolico.

```
C=L
0001            *042 341
0002  LOOPIT,   DCRC     /DECREMENT THE COUNT
0003            JNZ      /LOOP BACK IF THE COUNT
0004            LOOPIT   /IS NON-ZERO
0005            JMP      /THE COUNT IS ZERO, SO
0006            100      /JUMP TO 043 100 (HEXADECIMAL
0007            23H      /2340).
0008
0009
C=X
OCTAL
M OR T ? M
AUTO-LOAD ?N
ERRORS DETECTED = 000

NZ AT LINE # 01-0005

ERRORS DETECTED = 001
------


      TYCHON EDITOR-ASSEMBLER V-3                          PAGE 01-001


                        *042 341
042 341 015  LOOPIT,    DCRC     /DECREMENT THE COUNT
042 342 302             JNZ      /LOOP BACK IF THE COUNT
042 343 341             LOOPIT   /IS NON-ZERO

NZ AT LINE # 01-0005
042 344 303             JMP      /THE COUNT IS ZERO, SO
042 345 100             100      /JUMP TO 043 100 (HEXADECIMAL
042 346 043             23H      /2340).

------


      TYCHON EDITOR-ASSEMBLER V-3                          PAGE 01-002


LOOPIT =042 341
ERRORS DETECTED = 001
```

Dal momento che la seconda istruzione di salto (JMP) si vale di un numero ottale come secondo byte e di uno esadecimale come terzo byte, non è necessario nessuno zero. Questo è obbligatorio nelle istruzioni di tre byte solo nel caso in cui, come secondo byte dell'istruzione stessa, sia impiegato un indirizzo simbolico oppure una stringa definita mediante la pseudo-op DW.

Superamento dei Limiti di Memoria (memory exceeded (ME!))

Assemblando un programma memorizzato in memoria oppure su banda perforata, TEA procede a generare due tabelle: la tabella dei simboli degli indirizzi ed una seconda tabella contenente le stringhe ed i valori di otto bit assegnati alle stringhe in base alla pseudo-op DW. Questa seconda tabella ha il nome di tabella dei byte definiti. Per la memorizzazione di queste due tabelle è comunque necessario un adeguato spazio di memoria. Lo spazio di memoria occorrente sarà per l'appunto determinato in base al numero degli indirizzi simbolici definiti

Esempio 2-28. Determinazione dello spazio di memoria a disposizione della tabella degli indirizzi simbolici e di quella dei byte definiti.

```
INITIAL ADDRESS ? 0040 000
FINAL ADDRESS ? 0077 377
SOURCE CURRENTLY IN MEMORY ? Y
C=Q
052 365
077 377
LINE # =  0179
C=M
C=Q
2A F5
3F FF
LINE # =  0179
C=
```

all'interno del programma e dal numero di pseudo op DB e DW utilizzate all'*inizio* del programma.

La memoria di cui TEA si serve per queste due tabelle non fa parte della zona di memoria utilizzata dal text buffer, ma è *interna ai limiti di memoria* stabiliti dall'utente al momento di avviare TEA o di impiegare il comando M (MEMORY).

Nell'Esempio 2-28 l'indirizzo iniziale è posto a 040 000 (2000 hex) e quello finale a 077 377 (3FFF hex). Poichè in memoria si trova già memorizzato un programma sorgente, la risposta alla terza domanda è Y (YES). Servendosi del comando Q, può essere determinata l'estensione della memoria utilizzata dal buffer del testo. Come si può constatare, il text buffer è memorizzato nelle locazioni di memoria comprese tra 040 000 sino a 052 365 (da 2000 a 2AF5 hex). Ne deriva che tutta la parte di memoria che si estende da 052 365 sino a 077 377 (da 2AF5 sino a 3FFF hex) può essere utilizzata per la tabella dei simboli degli indirizzi e quella dei byte definiti. Essa è più che sufficente, dato che il programma consta di sole 178 linee di testo.

Quanto è grande lo spazio di memoria occupato da ciascun elemento di una tabella? Per un indirizzo simbolico o una stringa definita in base alla pseudo-op DW, a ciascun elemento della tabella dei simboli degli indirizzi occorre un numero di locazione pari a quello dei caratteri contenuti nella stringa più altre tre locazioni di memoria. Ne consegue che, allorchè si passa a definire LOOPIT mediante la pseudo-op DW, sono necessarie 6 + 3 (nove) locazioni di memoria per memorizzare i caratteri ASCII della stringa nonchè il valore di 16 bit ad assa associato. La locazione di memoria addizionale serve a memorizzare il valore 075 (3D hex), utilizzato come separatore della stringa dal valore di 16 bit. Per ogni stringa utilizzata in una pseudo-op DB, ogni carattere della stringa equivale ad una locazione di memoria; un'ulteriore locazione di memoria è necessaria a memorizzare il separatore 075 (3D hex); un'ultima locazione di memoria è poi necessaria per memorizzare il valore di 8 bit da assegnarsi insieme all'istruzione. Risulta dunque che in 1 K (1.024) locazioni di memoria R/W possono essere memorizzati 113 indirizzi simbolici di sei caratteri oppure stringhe definite mediante la pseudo-op DW. Nel caso di byte definiti, avendo a disposizione 1K di memoria R/W, possono essere memorizzate in memoria 128 stringhe di sei carattreri insieme ai valori ad esse assegnati.

Nell'Esempio 2-28, perciò, la memoria disponibile è sufficiente a memorizzare centinaia di indirizzi simbolici, stringhe di caratteri relative a parole definite e stringhe di caratteri relative a byte definiti.

Il messaggio di errore per superamento di memoria (ME!) si produce allorchè, mentre TEA sta procedendo a compilare la tabella degli indirizzi simbolici oppure quella dei byte definiti, la memoria a disposizione non è più sufficiente a memorizzare le tabelle al completo. Per mettere in evidenza questa condizione di errore, si può far riferimento ad uno dei brevi esempi già considerati.

Allorchè, nell'Esempio 2-29, si è dato il via a TEA, al buffer del testo sono riservate 2K locazioni di memoria (da 040 000 a 050 000 ottale, da 2000 a 2800 hex). Poichè in memoria è già presente un programma, si

Esempio 2-29. Generazione di un messaggio di errore ME! in seguito al cambiamento dei limiti in memoria.

```
INITIAL ADDRESS ? 0040 000
FINAL ADDRESS ? 0050 000
SOURCE CURRENTLY IN MEMORY ? Y
C=L
0001
0002            *042 341
0003  LOOPIT,  DCRC     /DECREMENT THE COUNT
0004            JNZ      /LOOP BACK IF THE COUNT
0005            LOOPIT   /IS NON-ZERO
0006            0
0007            RET
0008
C=Q
040 150
050 000
LINE # =  0008
C=X
OCTAL
M OR T ? M
AUTO-LOAD ?N
ERRORS DETECTED = 000

ERRORS DETECTED = 000
------
?
C=M
INITIAL ADDRESS ?
FINAL ADDRESS ? 0040 154
SOURCE CURRENTLY IN MEMORY ? Y
C=L
0001
0002            *042 341
0003  LOOPIT,  DCRC     /DECREMENT THE COUNT
0004            JNZ      /LOOP BACK IF THE COUNT
0005            LOOPIT   /IS NON-ZERO
0006            0
0007            RET
0008
C=Q
040 150
040 154
LINE # =  0008
C=X
OCTAL
M OR T ? M
AUTO-LOAD ?N
ME AT LINE # 01-0003
?
C=
```

introduce un Y (YES) e quindi si effettua il listing del programma con la telescrivente. Si introduce poi il comando Q in modo da poter determinare il numero di locazioni in memoria utilizzate dal buffer nonchè quello delle locazioni di memoria disponibili per le tabelle degli indirizzi simbolici e dei byte definiti. La memoria esistente è più che sufficente per la memorizzazione di queste tabelle, cosicchè il programma è regolarmente assemblato. Non appena ha inizio la terza passata, mentre TEA sta ancora stampando le lineette, si introduce un CTRL/C, riportando TEA nel modo di comando. Poichè non vi è stato il messaggio di errore ME!, si ha conferma che la memoria per le tabelle era più che sufficiente.

Si introduce allora il comando M (MEMORY), utilizzando il medesimo indirizzo iniziale di prima (040 000, 2000 hex). Si cambia invece l'indirizzo finale da 050 000 a 040 154 (da 2800 a 206C hex). Questo ha come conseguenza che le locazioni di memoria utilizzabili per le due tabelle che TEA deve generare si sono ridotte a quattro soltanto. Dopo che sono stati cambiati i limiti di memoria, si procede ad un listing del programma con lo scopo di accertarsi che esso sia ancora presente in memoria. A questo punto si procede ad assemblarlo.

Nel corso della prima passata, TEA deve porre la stringa LOOPIT, uno 075 ed un indirizzo di 16 bit (042 341, 22E1 hex) nella tabella degli indirizzi simbolici. Per questo, tuttavia, occorre un numero di locazioni di memoria maggiore delle quattro "lasciate", cosicchè TEA stampa il messaggio di errore ME! AT LINE #01-003. Ciò corrisponde a ciò che si poteva prevedere, in quanto al momento in cui si manifesta l'errore TEA sta "elaborando" l'indirizzo simbolico LOOPIT. L'errore di memoria superata (ME!) corrisponde a quello che si dice un *errore irrimediabile* (in genere indicato come FATAL ERROR). Il significato di questo termine sta nel fatto che, nel caso che avvenga un tale genere di errore di assemblaggio, TEA arresta immediatamente il procedimento di assemblaggio e ritorna al modo di comando, come si può constatare dall'esame dell'ultima porzione dell'Esempio 2-29. Dopo la stampa del messaggio di errore ME! l'assembler si astiene dall'eseguire qualunque ulteriore passata a qualunque altra azione, limitandosi a tornare al modo di comando. Tutte le condizioni di errore descritte in precedenza sono *rimediabili (non fatali)*; il procedimento di assemblaggio può proseguire anche se vi è stata la scoperta di un errore. In seguito a questa circostanza, è possibile localizzare ulteriore errori presenti nel resto del programma. Altri tipi di errori non fatal e fatal saranno visti tra breve.

Nel caso che si verifichi il messaggio di errore per memoria superata (ME!), le possibili soluzioni sono tre. Una di esse consiste nell'introdurre il comando M (MEMORY) ed aumentare l'estensione della memoria adibita al buffer del testo. Se la memoria R/W a disposizione è già stata tutta riservata al text buffer, si può provare a ridurre il numero degli indirizzi simbolici, delle parole e dei byte definiti sino a che il messaggio di errore ME! non si verifichi più. Se questo secondo sistema non è attuabile si può sempre assemblare il programma da banda perforata (oppure da un qualsiasi altro supporto dal quale il dispositivo IOR possa leggere una linea di testo per volta). Il metodo utilizzato per assemblare

programmi da banda perforata sarà descritto in un'altra sezione di questo stesso capitolo.

In questo libro, si è precedentemente affermato che il text buffer abbia a sua disposizione la totalità della memoria riservata dall'utente nel corso del dialogo di inizializzazione con TEA oppure mediante il comando M (MEMORY). Si può osservare, giunti a questo punto, che ciò non sempre è vero. Se, infatti, si deve assemblare un programma memorizzato interamente in memoria, vi deve essere abbastanza memoria *non* utilizzata dal buffer affinchè TEA possa servirsene per la tabella dei simboli degli indirizzi e per quella dei byte definiti. Soltanto nel caso che ci si valga dell'editor di TEA per redigere un manoscritto, un modulo postale o qualche altro genere di informazione che poi TEA non debba assemblare, l'intero buffer può essere utilizzato per memorizzare del testo.

Errori di Definizione (DE)

Un errore di definizione ha luogo allorchè si impiega la pseudo-op DW o quella DB in qualsiasi punto fuorchè all'inizio di un programma. Come già accennato, queste pseudo-op devono essere utilizzate prima che nel programma compaia qualunque simbolo mnemonico di istruzione, indirizzo simbolico definito o indirizzo definito (*). Nel caso che le pseudo-op DW o DB siano utilizzate in qualunque altro punto, compare un DE AT LINE # xx-nnnn. Poichè si tratta di un fatal error, TEA arresta le operazioni di assemblaggio e ritorna immediatamente al modo di comando. Nel caso che si verifichi questo genere di errore, le pseudo-op DW o DB responsabili *devono* essere spostate all'inizio del programma per permettere l'esecuzione dell'assemblaggio.

Nell'esempio 2-30 compare in primo luogo il listing del programma da assemblare. Come si può constatare, la pseudo-op DW è utilizzata alla linea uno e quella DB alla linea nove e dieci posposta ad alcuni simboli di istruzione, a un indirizzo simbolico definito (START) e ad un indirizzo definito (*203 013). Si introduce quindi il comando X (ASSEMBLE), senza che nel corso della prima passata sia scoperto alcun errore. Nella seconda passata, tuttavia, TEA stampa il messaggio di errore DE AT LINE # 01-0009, a causa della linea che contine DB CR 015. Poichè si tratta di un fatal error, TEA ritorna immediatamente al modo di comando. In conseguenza di ciò, nel corso della seconda passata dell'assembler, sarà impossibile arrivare alla linea 10.

Si cancellano allora le linee da 9 a 11 e si impiegano le pseudo-op DB alle linee due e tre per l'assegnazione di valori di 8 bit a CR ed LF. Tali modifiche risultano evidenti dal listing del programma. Si introduce, quindi, di nuovo il comando X (ASSEMBLE) e l'operazione di assemblaggio può allora svolgersi senza errori, dal momento che tutte le pseudo-op DW e DB sono ora poste prima di tutti i simboli di istruzione, nonchè di tutti gli indirizzi simbolici definiti e degli indirizzi definiti. Si badi che il procedimento di assemblaggio è stato bloccato alla fine della seconda passata, ma che, ad onta di ciò, si può star certi che l'eventuale

Esempio 2-30. Un errore di definizione (DE) causato dall'impiego di due pseudo-op DB in un punto sbagliato del programma.

```
C=L
0001            DW TTYOUT 001 034
0002
0003            *203 013
0004  START,    CALL     /PRINT A CARRIAGE RETURN AND A
0005            CRLF     /LINE FEED ON THE TELETYPEWRITER
0006            0
0007            HLT      /THEN HALT
0008
0009            DB CR 015
0010            DB LF 0AH
0011
0012  CRLF,     MVIA     /LOAD THE A REGISTER WITH
0013            CR       /THE ASCII VALUE FOR A
0014            CALL     /CARRIAGE RETURN AND THEN
0015            TTYOUT   /PRINT IT ON THE
0016            0        /TELETYPEWRITER
0017            MVIA     /THEN LOAD THE A REGISTER WITH
0018            LF       /THE ASCII VALUE FOR A LINE FEED
0019            JMP      /AND PRINT IT ON THE
0020            TTYOUT   /TELETYPEWRITER
0021            0
0022
0023
C=X
OCTAL
M OR T ? M
AUTO-LOAD ?N
ERRORS DETECTED = 000

DE AT LINE # 01-0009
?
C=D9-11

C=I2
0002            DB CR 015
0003            DB LF 0AH
0004  ?
C=L
0001            DW TTYOUT 001 034
0002            DB CR 015
0003            DB LF 0AH
0004
0005            *203 013
0006  START,    CALL     /PRINT A CARRIAGE RETURN AND A
0007            CRLF     /LINE FEED ON THE TELETYPEWRITER
0008            0
0009            HLT      /THEN HALT
0010
0011  CRLF,     MVIA     /LOAD THE A REGISTER WITH
0012            CR       /THE ASCII VALUE FOR A
0013            CALL     /CARRIAGE RETURN AND THEN
0014            TTYOUT   /PRINT IT ON THE
0015            0        /TELETYPEWRITER
0016            MVIA     /THEN LOAD THE A REGISTER WITH
0017            LF       /THE ASCII VALUE FOR A LINE FEED
0018            JMP      /AND PRINT IT ON THE
0019            TTYOUT   /TELETYPEWRITER
0020            0
0021
0022
C=X
OCTAL
M OR T ? M
AUTO-LOAD ?N
ERRORS DETECTED = 000

ERRORS DETECTED = 000
------?
C=
```

terza passata, ma che, ad onta di ciò, si può star certi che l'eventuale terza passata non farebbe scoprire nessun errore, per il semplice motivo che tutti gli errori che si manifestano durante la terza passata sono gli stessi rivelati nel corso della seconda .

Il programma dell'Esempio 2-30 potrebbe anche essere modificato in modo da far comparire la pseudo-op DW in un punto non adatto. L'insegnamento che se ne potrebbe trarre non aggiungerebbe però molto a quanto già si conosce, in quanto TEA genera un messaggio di errore DE nel corso della seconda passata dell'assembler *tanto* che sia la pseudo-op DB ad essere utilizzata in un punto non corretto quanto che si tratti della pseudo-op DW.

No Address (NA) (assenza dell'indirizzo)

Se mai ci si dimentica di definire almeno un indirizzo del programma, TEA genera il messaggio di errore NA AT LINE # nn-xxxx, significando così che non conosce quali indirizzi deve utilizzare per le varie linee di testo.

Nell'Esempio 2-31 è compilata ed assemblata la subroutine di tre istruzioni. Al termine della prima passata, TEA stampa ERRORS DETECTED = 000 e comincia quindi ad eseguire la seconda passata attraverso il text buffer. Avendo trovato, tuttavia, un simbolo di istru-

Esempio 2-31. Messaggio di errore NA generato nel caso che in un programma non siano stati definiti gli indirizzi.

```
C=L
0001  LOOPIT, DCRC     /DECREMENT THE COUNT
0002          JNZ      /LOOP BACK IF THE COUNT
0003          LOOPIT   /IS NON-ZERO
0004          0
0005          RET
0006
C=X
OCTAL
M OR T ? M
AUTO-LOAD ?N
ERRORS DETECTED = 000

NA AT LINE # 01-0001
?
C=I 1
0001          *010 000
0002  ?
C=L 1-3
0001          *010 000
0002  LOOPIT, DCRC     /DECREMENT THE COUNT
0003          JNZ      /LOOP BACK IF THE COUNT
C=X
OCTAL
M OR T ? M
AUTO-LOAD ?N
ERRORS DETECTED = 000

ERRORS DETECTED = 000
------

?
C=
```

zione nella prima linea di testo non sa quale indirizzo debba essere associato al codice del simbolo. Si ricordi, a questo proposito, che, allorchè si perfora su banda un programma oggetto, l'informazione relativa all'indirizzo deve sempre accompagnare i valori corrispondenti alle istruzioni e ai dati. Tale informazione sugli indirizzi deve essere sempre presente affinchè, rileggendo la banda perforata nel microcalcolatore, l'8080 possa determinare dove memorizzare il programma.

Con la stampa del messaggio di errore NA AT LINE #01-0001, TEA segnala che occorre definire un indirizzo all'inizio della linea 0001. Nel caso che le prime due o tre linee del buffer del testo fossero commenti e la linea cinque contenesse il primo simbolo di istruzione, il messaggio di errore sarebbe NA AT LINE # 01-0005 poichè ai commenti non sono assegnate locazioni di memoria, così come, d'altronde, essi non sono perforati sulla banda del programma oggetto o caricati in memoria nel caso in cui ci si serve della funzione di auto-caricamento.

L'errore NA appartiene alla categoria degli errori fatali. TEA ritorna perciò al modo di comando subito dopo aver stampato il messaggio di errore. Comunque, nell'esempio 2-31, all'inizio della linea uno si inserisce (I1!) un indirizzo definito. Allorchè, dopo questa modifica, il programma è assemblato, non si manifestano più errori in quanto TEA è ora edotto su dove debbano essere memorizzati tutti i valori relativi alle istruzioni ed ai dati. Il procedimento di assemblaggio è interrotto di nuovo al termine della seconda passata mediante l'introduzione di un CTRL/C.

Overlap Error (OL) (errore di sovrapposizione)

Il messaggio di errore di sovrapposizione (OL) si verifica soltanto nel caso che, avendo risposto Y (YES) alla domanda "AUTO-LOAD?" che segue l'introduzione del comando X (ASSEMBLE), TEA scopra che le informazioni contenute nella memoria R/W andrebbero perdute qualora il codice di istruzione o di dato della linea di testo appena assemblata fosse trascritto in memoria.

Si supponga di assemblare la subroutine elencata nell'Esempio 2-31 e che la risposta alla domanda AUTO-LOAD? sia Y (YES). Se l'assemblaggio del programma avviene senza errori supponendo che al buffer del testo sia già stato aggiunto l'indirizzo definito, la subroutine sarà caricata in memoria a partire da 010 000. Dopo che il programma è stato assemblato nel corso della seconda passata, si potrà, perciò, utilizzare un monitor di sistema o un debugger per esaminare un pò di locazioni di memoria cominciando così da quella 010 000. Nel caso si tratti di locazioni di memoria R/W, sarebbe lecito attendersi di trovare il codice corrispondente al simbolo DCRC (015 ottale, 0D hex) seguito dal codice corrispondente al simbolo di istruzione JNZ, l'indirizzo di 16 bit di LOOPIT ed il codice corrispondente all'istruzione RET (302, 000, 010 e 311 ottale, C2, 00, 08 e C9 hex).

Caricando in memoria R/W dei codici di 8 bit nel corso della seconda

passata dell'assembler, occorre cautelarsi contro un errore tipico di questa procedura. Può darsi infatti che i valori relativi ai dati siano trascritti nelle locazioni di memoria R/W adibite alla memorizzazione del programma TEA, del text buffer, della tabella dei simboli degli indirizzi, di quella dei byte definiti o nelle locazioni riservate allo stack e ai dati temporanei.

Per accertarsi che TEA od una qualsiasi di queste altre locazioni di memoria R/W non siano "intasati" dalla funzione di auto-caricamento, TEA confronta l'indirizzo al quale deve memorizzare il codice con: (1) il primo e l'ultimo indirizzo di memoria adibiti alla memorizzazione del programma TEA, (2) la prima ed ultima locazione di memoria utilizzata dallo stack o come memoria temporanea, (3) l'indirizzo iniziale del buffer del testo (fissato dall'utente) e l'ultimo indirizzo utilizzato dalla tabella dei simboli degli indirizzi (la tabella dei simboli degli indirizzi è "costruita" in memoria subito di seguito alle linee di testo del buffer; la tabella dei simboli degli indirizzi ed il testo possono essere considerati come costituenti una sezione continua di memoria utilizzata da TEA) e (4) la prima ed ultima locazione di memoria adibite alla tabella dei byte definiti. Quest'ultima tabella si trova al confine della memoria R/W disponibile; essa cresce verso il basso, a partire dall'indirizzo finale specificato dall'utente nel dialogo iniziale con TEA. Nel caso ci si preoccupi che la tabella dei byte definiti invada quella dei simboli degli indirizzi, e *viceversa*, TEA provvede a controllare che questo non accada. Altrimenti nel corso del procedimento di assemblaggio si ottiene un messaggio di errore ME!.

In conclusione, prima di riporre in memoria R/W un qualunque valore, in base alla funzione di auto-caricamento di TEA nel corso della seconda passata del procedimento di assemblaggio, TEA effettua innanzitutto un confronto di tutti questi indirizzi limite con l'indirizzo di memoria in corrispondenza del quale deve essere caricato il valore di 8 bit. Nel caso che l'indirizzo di memoria sia compreso comunque entro questi indirizzi, TEA provvede a stampare il messaggio di errore OL, insieme al numero di linea in corrispondenza del quale si "è verificata" la sovrapposizione. In caso di errore di sovrapposizione, la funzione di auto-caricamento è disabilitata automaticamente, evitando così che insieme ad ogni linea del testo assemblato sia stampato un messaggio di errore OL. Naturalmente, se, in un momento successivo, si dà nuovamente il via al procedimento di assemblaggio, TEA ripete ancora la domanda "AUTO-LOAD?", in modo che, se non vi è sovvrapposizione, il programma può essere caricato direttamente in memoria nel corso della seconda passata dell'assembler. È naturale che, prima, si deve cambiare un indirizzo definito (*) in modo che tale errore non si verifichi più.

L'Esempio 2-32 riporta un breve programma che è stato caricato automaticamente in memoria nel corso della seconda passata. Come prima cosa si effettua il listing del programma. Si introduce poi il comando Q (QUERY) per scoprire quali siano le sezioni di memoria utilizzate e quali quelle a disposizione per la memorizzazione del pro-

Esempio 2-32. Caricamento automatico in memoria di una versione oggetto del programma assemblato nel corso della seconda passata dell'assembler.

```
C=L
0001            *050 000
0002  LOOPIT,   DCRC     /DECREMENT THE COUNT
0003            JNZ      /JUMP BACK IF THE COUNT
0004            LOOPIT   /IS NON-ZERO
0005            0
0006            RET      /RETURN WHEN THE COUNT IS ZERO
0007
C=Q
040 206
077 377
LINE # =   0007
C=X
OCTAL
M OR T ? M
AUTO-LOAD ?Y
ERRORS DETECTED = 000

ERRORS DETECTED = 000
------


      TYCHON EDITOR-ASSEMBLER V-3                          PAGE 01-001


                        *050 000
050 000 015  LOOPIT,    DCRC     /DECREMENT THE COUNT
050 001 302             JNZ      /JUMP BACK IF THE COUNT
050 002 000             LOOPIT   /IS NON-ZERO
050 003 050             0
050 004 311             RET      /RETURN WHEN THE COUNT IS ZERO



?
C=Z
ENTER A COMMAND, THEN A CR.
COMMAND = DBUG

?
050 000 / 015
050 001 / 302
050 002 / 000
050 003 / 050
050 004 / 311
```

gramma oggetto. Poichè il programma non contiene pseudo-op DW nè DB ed esiste un unico indirizzo simbolico definito, la tabella dei simboli degli indirizzi risulta molto breve. TEA, perciò, non dovrebbe incontrare nessuna difficoltà per caricare automaticamente in memoria il programma, a partire dall'indirizzo 050 000 (2800 hex).

Si inizia quindi il procedimento di assemblaggio e l'utente specifica che il programma deve essere caricato automaticamente nella memoria R/W (nel corso della seconda passata). Nel corso della prima e seconda passata non si danno errori di nessun genere e il listing della terza passata permette all'utente di constatare personalmente dove sono memorizzati in memoria i valori di 8 bit. L'utente ha introdotto il comando Z, facendo avvenire il trasferimento del controllo ad un programma monitor di sistema in cui è stato introdotto il nome che distingue il prossimo programma in ordine di esecuzione (DBUG; un debugger rivolto al

Esempio 2-33. Cambiamento dell'indirizzo definito, in seguito al quale si manifesta un errore di sovrapposizione (OL) nel corso della seconda passata dell'assembler.

```
C=L
0001            *040 100
0002  LOOPIT,   DCRC     /DECREMENT THE COUNT
0003            JNZ      /JUMP BACK IF THE COUNT
0004            LOOPIT   /IS NON-ZERO
0005            0
0006            RET      /RETURN WHEN THE COUNT IS ZERO
0007
C=Q
040 206
077 377
LINE # =  0007
C=X
OCTAL
M OR T ? M
AUTO-LOAD ?Y
ERRORS DETECTED = 000
0
OL AT LINE # 01-0002
 0
ERRORS DETECTED = 001
------


      TYCHON EDITOR-ASSEMBLER V-3                          PAGE 01-001


                        *040 100
040 100 015  LOOPIT,    DCRC     /DECREMENT THE COUNT
040 101 302             JNZ      /JUMP BACK IF THE COUNT
040 102 100             LOOPIT   /IS NON-ZERO
040 103 040             0
040 104 311             RET      /RETURN WHEN THE COUNT IS ZERO


?
C=Z
ENTER A COMMAND, THEN A CR.
COMMAND = DBUG

?
040 100 / 124
040 101 / 110
040 102 / 105
040 103 / 040
040 104 / 103
```

linguaggio assembler). L'uso di DBUG[1] permette di esaminare la sezione di memoria R/W dove è memorizzato il programma, che, come previsto, è stato caricato in memoria correttamente. Tale circostanza è dimostrata dalla corrispondenza tra il contenuto della memoria che si osserva grazie a DBUG ed i numeri ottali riportati nel listing. A questo proposito non è tanto essenziale conoscere il modo di operare di questo monitor di sistema o DBUG, quanto invece è importante rendersi conto di ciò che la funzione di auto-caricamento nonchè dei suoi pregi e difetti.

Uno dei vantaggi più evidenti consiste nel fatto che per dar corso all'esecuzione del programma in linguaggio assembler, non vi è più bisogno di rileggere nel calcolatore una banda perforata o un'audiocassetta con il programma oggetto. Un cambiamento dell'indirizzo definito posto all'inizio della subroutine può dar luogo all'errore di sovrap-

posizione (OL). Dato che, come dimostra il comando Q (QUERY) impiegato nell'Esempio 2-32, le sei linee di testo sono memorizzate nelle locazioni di memoria da 040 000 a 040 206 (da 2000 a 2086 hex), il cambiamento in 040 100 (2040) dell'indirizzo definito posto all'inizio della subroutine potrebbe causare un errore di sovrapposizione. Nell'Esempio 2-33 l'indirizzo definito è stato, per l'appunto, cambiato in tal modo. Dopo il listing del programma e l'introduzione dei comandi Q (QUERY) ed X (ASSEMBLE), alla domanda "AUTO-LOAD?" l'utente risponde con Y (YES).

L'esecuzione della prima passata è così avvenuta senza che si manifestassero errori. Non così per la seconda passata: allorchè TEA si è accinto a riporre il codice corrispondente al simbolo DCRC nella locazione di memoria 040 100, si è manifestato l'errore di sovrapposizione. Come prevedibile, il numero di linea elencato nel messaggio di errore fa riferimento alla prima linea contenente un simbolo di istruzione. Se ne deduce che la semplice definizione dell'indirizzo 040 100 non comporta, di per se, il manifestarsi di un errore; prima che quest'ultimo si manifesti occorre che TEA si accinga realmente a riporre un qualche valore in quella locazione di memoria.

Si è già avuto occasione di accennare a come la tabella dei simboli degli indirizzi generata da TEA sia memorizzata subito di seguito al text buffer. È possibile dimostrare che TEA memorizza "qualcosa" dopo le linee di testo "sfruttando" l'errore di sovrapposizione. Nell'Esempio 2-34 l'indirizzo definito è stato cambiato in 040 215, superiore a quello dell'ultima locazione di memoria utilizzata dal text buffer come risulta dal comando Q (QUERY). Anche questo indirizzo, tuttavia, porta TEA

Esempio 2-34. Rivelazione della presenza della tabella dei simboli degli indirizzi per mezzo del messaggio di errore di sovrapposizione (OL).

```
C=Q
040 206
077 377
LINE # =   0007
C=C1
0001              *040 215
0002  ?
C=X
OCTAL
M OR T ? M
AUTO-LOAD ?Y
ERRORS DETECTED = 000
0
OL AT LINE # 01-0002
 ?
C=C1
0001              *040 225
0002  ?
C=X
OCTAL
M OR T ? M
AUTO-LOAD ?Y
ERRORS DETECTED = 000
00
ERRORS DETECTED = 000
```

Esempio 2-35. Prove di assemblaggio di un programma contenente delle stringhe di caratteri non definite.

```
C=L
0001            *050 000
0002  START,    LXISP    /LOAD THE STACK POINTER WITH
0003            STACK    /A R/W MEMORY ADDRESS
0004            0
0005            CALL     /GET A CHARACTER FROM THE
0006            TTYIN    /TELETYPEWRITER OR CRT
0007            0
0008            MOVZA    /SAVE THE ASCII CHARACTER IN C
0009            INRS     /INCREMENT THE D REGISTER
0010            JNZ      /IF NON-ZERO, GET ANOTHER
0011            LOOP     /CHARACTER
0012            0
0013
C=X
OCTAL
M OR T ? M
AUTO-LOAD ?N
ERRORS DETECTED = 000

UD AT LINE # 01-0003

UD AT LINE # 01-0006

UD AT LINE # 01-0008

UD AT LINE # 01-0009

UD AT LINE # 01-0011

ERRORS DETECTED = 005
------

      TYCHON EDITOR-ASSEMBLER V-3                          PAGE 01-001

                        *050 000
050 000 061  START,     LXISP    /LOAD THE STACK POINTER WITH

UD AT LINE # 01-0003
050 001 000             STACK    /A R/W MEMORY ADDRESS
050 002 000             0
050 003 315             CALL     /GET A CHARACTER FROM THE

UD AT LINE # 01-0006
050 004 000             TTYIN    /TELETYPEWRITER OR CRT
050 005 000             0

UD AT LINE # 01-0008
050 006 107             MOVZA    /SAVE THE ASCII CHARACTER IN C

UD AT LINE # 01-0009
050 007 004             INRS     /INCREMENT THE D REGISTER
050 010 302             JNZ      /IF NON-ZERO, GET ANOTHER

UD AT LINE # 01-0011
050 011 000             LOOP     /CHARACTER.
050 012 000             0

   ------

      TYCHON EDITOR-ASSEMBLER V-3                          PAGE 01-002

START  =050 000
ERRORS DETECTED = 005
```

a generare il messaggio di errore OL. L'incremento a 040 225 dell'indirizzo di memoria fa sì che l'operazione di assemblaggio sia eseguita senza errori. Incrementi e decrementi successivi del valore dell'indirizzo definito consentono di scoprire la prima locazione di memoria disponibile dopo la tabella dei simboli degli indirizzi.

Non Definito (Undefined) (UD)

Nel caso che un errore non ricada nel dominio delle altre condizioni di errore, TEA procede a generare il messaggio di errore indefinito (UD). Analogamente a quanto avviene per gli altri messaggi di errore; TEA provvede anche a stampare il numero di blocco ed il numero di linea dove si trova memorizzata la stringa di caratteri che risulta non definita. La frequenza con cui ci si imbatte in questo messaggio di errore è normalmente superiore a quella di tutti gli altri.

Nell'Esempio 2-35 compaiono alcune stringhe di caratteri non definite, corrispondenti ad altrettanti errori rilevati nel corso della seconda a terza passata di TEA. Come ben si può prevedere, la causa di tali messaggi di errore UD sono le stringhe STACK, TTYIN, MOVZA, INRS e LOOP non definite. Gli indirizzi simbolici dovrebbero essere stati definiti come indirizzi simbolici internamente al programma oppure per mezzo della pseudo-op DW. È lecito pensare che le istruzioni MOVZA ed INRS siano dovute ad un banale errore, per il semplice motivo che l'8080 prevede dei simboli di istruzione quasi uguali. L'editor di TEA può quindi essere utilizzato per mutare tali simboli in MOVCA ed INRD, come giustificherebbero i commenti contenuti nel programma.

Può accadere che i messaggi di errore prodotti da TEA sembrino privi di coerenza. Se, ad esempio, si fa uso del simbolo di istruzione JNX, TEA provvede a generare un messaggio di errore BN in luogo di quello UD che ci si potrebbe attendere. La spiegazione di questa anomalia è da ricercarsi nell'organizzazione interna di TEA. Se esso non riesce a trovare il simbolo JNX nella tabella dei simboli permanenti (nella quale sono memorizzati tutti i simboli validi di istruzione dell'8080, passa a controllare che non si tratti di un indirizzo simbolico valido (da qui la possibilità di creare in un programma una tabella degli indirizzi simbologici). Nel caso che non si tratti nemmeno di un indirizzo simbolico, TEA provvede a verificare che la stringa non sia per caso stata definita mediante la pseudo-op DB e, in caso negativo, che non si tratti di un numero ottale o esadecimale valido. Poichè la stringa non lo è, il risultato finale è un messaggio di errore BN. L'importante non è, dunque, di preoccuparsi che il messaggio di errore abbia un significato inequivocabile, ma, piuttosto, di cercare si stabilire le cause prima dell'errore. In ordine di probabilità, può darsi che sia stato compiuto un errore di battitura; che per il simbolo di istruzione l'indirizzo definito, il commento o l'indirizzo simbolico non sia stato utilizzato il formato corretto già specificato in altra parte di questo stesso capitolo; oppure che ci si sia dimenticati di definire una stringa per mezzo della pseudo-op DB o DW.

ASSEMBLAGGIO DI UN PROGRAMMA MEMORIZZATO IN MEMORIA R/W

In molti degli esempi già riportati si è potuto osservare come l'impiego di TEA permetta di assemblare programmi memorizzati in memoria R/W (text buffer). I passi che, in generale, bisogna effettuare per assemblare un programma memorizzato interamente in memoria R/W possono essere riassunti come segue.

(1) Carica TEA in memoria R/W ed avvia la sua esecuzione. Nel caso che esso sia già memorizzato in memoria EPROM sarà sufficente farlo partire. Introduci un indirizzo iniziale ed uno finale per il buffer in ottale o esadecimale. Se il programma sorgente è unico e deve essere memorizzato in memoria R/W tutto in una volta, riserva al text buffer quanto più spazio di memoria R/W sia possibile. Crea il programma sorgente valendoti della tastiera della telescrivente o del CRT (IONOEC), o trascrivilo in TEA servendoti del comando R (READER: lettore) del dispositivo IOR, seguito da uno dei comandi A (APPEND), I (INSERT) o C (CHANGE).

(2) Allorchè il programma è stato editato in modo soddisfacente, riponilo nel dispositivo IOP che può essere un registratore da cassette o una perforatrice di banda. Questa precauzione garantisce di poter rileggere di nuovo in TEA il programma sorgente nell'eventualità che la sua copia in memoria risulti, per un qualsiasi motivo, distrutta o alterata.

(3) Una volta messo al sicuro il programma sorgente nel dispositivo IOP, introduci il comando X (ASSEMBLE), al quale TEA risponde stampando OCTAL oppure HEXADECIMAL, seguito dalla domanda "M OR T?".

Le parole OCTAL e HEXADECIMAL denotano il sistema di numerazione che sarà utilizzato nel generare il listing corrispondente alla terza passata. Nel caso si voglia effettuare la conversione dall'uno all'altro dei due sistemi, è sufficiente introdurre un CRTL/C e, quindi, una volta che TEA è tornato al modo di comando, farlo seguire dal comando O (OCTAL) oppure H (HEXADECIMAL) per reintrodurre, da ultimo, il comando X (ASSEMBLE).

TEA stamperà una seconda volta il sistema di numerazione da utilizzare nel listing della terza passata e formulerà la domanda "M OR T?". Dal momento che il programma che si desidera assemblare è memorizzato interamente in memoria, deve essere introdotta un M. TEA chiederà a questo punto se la versione oggetto del programma dovrà essere caricata automaticamente in memoria nel corso della seconda passata (AUTO-LOAD?). Dopo l'introduzione di N (NO) oppure Y (YES), TEA comincerà il procedimento di assemble.

Dopo aver creato la tabella dei simboli degli indirizzi nel corso della prima passata, TEA provvede a stampare "ERRORS DETECTED" (Errori rilevati), insieme al numero degli eventuali errori di definizione, passando quindi ad effettuare la seconda passata attraverso il programma sorgente.

Nel corso di questa passata TEA provvede ad inviare in uscita verso il

dispositivo IOP (perforatrice di I/O) la versione oggetto del programma e, se necessario, a caricarla in memoria. Nel caso che TEA compia quest'ultima operazione, esso non procede a nessun tipo di controllo diretto ad accertare che il programma non sia riposto in un'area di memoria inesistente oppure a sola lettura. Il suo lavoro si limita a controllare che non si verifichi sovrapposizione.

Se, in funzione di dispositivo di I/O sia come perforatrice (IOP) che come unità di uscita (IOO), si utilizza una telescrivente, può accadere che durante la perforazione del nastro siano stampati dei caratteri privi di senso compiuto. Tali caratteri non sono *significativi* e la loro presenza è giustificata unicamente dal fatto che alcuni codici *binari* della versione oggetto del programma corrispondono a dei caratteri ASCII compresi tra quelli che la stampante è in grado di stampare. Se si esaminano con attenzione gli esempi di programma riportati in questo capitolo, illustranti l'impiego di TEA per assemblare il programma già presente in memoria, alcuni di tali caratteri sono stampati di seguito al primo messaggio "ERRORS DETECTED" generato da TEA al termine della prima passata.

(4) Nel caso che nel corso della seconda passata siano rilevati degli errori, questi saranno posti in evidenza mediante i messaggi di errore che sono stati appena trattati. Tali messaggi di errore saranno inviati al dispositivo di INO di uscita (IOO) nel corso del procedimento di assemblaggio. Una volta che il messaggio "ERRORS DETECTED" è stato inviato in uscita al termine della seconda passata, è possibile introdurre un CTRL/C nel caso che si siano manifestati degli errori in modo che, facéndo ritorno al modo di comando, TEA possa essere utilizzato per correggerli, qualunque sia la loro natura. Una volta corretti gli errori, è necessario riporre sul dispositivo perforatore di I/O (IOP) la nuova versione del programma sorgente. È bene ricordare che gli errori rilevati consistono in errori sintattici e non in errori di logica o di "flusso del programma" per cui è possibile che la ricerca e correzzione degli errori contenuti nel programma richieda l'ulteriore impiego di un debugger.

(5) Continua a servirti dell'editor e dell'assembler sino ad ottenere un programma assemblato esente da errori come si può stabilire in base al messaggio di errore che TEA stampa al termine della seconda passata del procedimento di assemblaggio.

Non appena tale procedimento si è compiuto senza errori, ferma TEA con un CTRL/C per tornare nel modo di comando. A questo punto, è necessario mettere in funzione o abilitare il dispositivo perforatore di I/O (IOP) che deve essere utilizzato per memorizzare la versione oggetto del programma (sorgente). Tale dispositivo può consistere nella perforatrice di banda della telescrivente o in registratore a cassette nel modo di funzionamento RECORD (registra). Dopo di ciò, fai ripartire con il comando X (ASSEMBLE) il procedimento di assemblaggio. L'esecuzione di questo passo è necessaria solamente se si desidera accantonare su di un dispositivo periferico la versione oggetto del programma.

(6) Una volta ottenuto un programma assemblato senza errori e fatto ripartire l'assembler, la versione oggetto del programma è posta in uscita

verso il dispositivo perforatore di I/O (IOP) nel corso della seconda passata. Dopo che sul dispositivo di INO (IOO) è stato stampato il messaggio di errore che conclude la seconda passata, la perforatrice di banda o il registratore a cassette possono anche essere esclusi. Il listing del programma assemblato sarà generato nel corso della terza passata e inviato al dispositivo di uscita di I/O (IOO). Una volta stampati il listing del programma assemblato ed il contenuto ordinato della tabella dei simboli degli indirizzi, TEA fa ritorno al modo di comando. È possibile, a questo punto caricare in memoria la versione oggetto del programma (solamente nel caso che a questo non abbia già provveduto la funzione di caricamento automatico) per procedere quindi alla sua esecuzione. Volendo caricare in memoria la versione oggetto del programma, occorre posizionare l'estremità iniziale della banda nel lettore di banda oppure riavvolgere la cassetta registrata, introdurre il comando B (BOOTSTRAP) e quindi avviare il lettore di banda o il registratore. Terminata del tutto la lettura in memoria della versione oggetto del programma si può introdurre il comando Z (EXIT) in modo che l'8080 esegua un programma monitor di sistema o un debbuger. Sia l'uno che l'altro consentono di cominciare l'effettiva esecuzione della versione oggetto del programma.

Nel caso che nel programma vi siano da apportare delle correzioni, si può avviare un'altra volta TEA e, se il text buffer è "intatto", eseguire ulteriori operazioni con l'editor e quindi assemblare di nuovo il programma. Sfruttando alternativamente le possibilità insite in TEA e nel monitor di sistema o nel debugger, si dovrebbe giungere in breve a creare un programma privo di errori assemblabile ed eseguibile correttamente sul microcalcolatore.

OSSERVAZIONE: Si osservi che, nel caricamento in memoria di un programma oggetto in base all'impiego del comando B (BOOTSTRAP), non è compiuto nessuno dei passi di confronto per sovrapposizione proprio della funzione di auto-caricamento. Ne discende che, caricando in memoria da banda un programma oggetto, questo può essere trascritto sopra le parti del tet buffer oppure in locazioni di memoria temporanea e area di stack riservate a TEA. Non è invece importante se il programma pggetto si trova ad essere trascritto sopra la tabella dei simboli degli indirizzi o a quella dei byte definiti, per il semplice motivo che queste tabelle sono generate ogniqualvolta si esegue la sezione di assembler di TEA. L'impiego del comando B (BOOTSTRAP) per il caricamento in memoria di programmi oggetto richiede perciò le dovute cautele.

ASSEMBLAGGIO DI UN PROGRAMMA MEMORIZZATO SU DI UN DISPOSITIVO DI I/O PERFORATORE (IOP)

Può avvenire che, ad un certo punto, il programma sorgente sia talmente cresciuto da non poter più essere contenuto tutto insieme in memoria R/W. Si rammenti come, nel corso del procedimento di assemblaggio, debba esservi uno sapzio di memoria sufficiente a che TEA possa formare la tabella dei simboli degli indirizzi nonchè quella dei byte

definiti.

(1) Carica TEA nella memoria R/W e comincia la sua esecuzione. Nel caso che TEA sia già memorizzato in memoria EPROM, è sufficiente dargli il via. Introduci un indirizzo iniziale ed un indirizzo finale del programma in forma ottale oppure esadecimale. Al text buffer è necessario riservare quanta più memoria è possibile. Crea il programma sorgente servendoti della tastiera della telescrivente o del CRT (IONOEC), oppure leggi in TEA il programma sorgente valendoti del comando R (READER) del dispositivo IOR, seguito da uno dei comandi A (APPEND), I (INSERT) o C (CHANGE).

Dato che il programma sorgente è troppo lungo per poter essere contenuto tutto insieme in memoria, esso deve essere spezzato in blocchi comprendenti dalle 100 alle 200 linee di testo, che saranno riposti sul dispositivo di perforazione I/O (IOP) valendosi del comando P (PUNCH). Si ricordi che questo dispositivo non può consistere in un registratore a cassetta in quanto, in tal caso, la lettura del testo nell'assembler dovrebbe essere effettuata a una sola riga per volta.

Normalmente, non è possibile sapere in anticipo se il programma sorgente è troppo grande per trovare in memoria spazio sufficiente che lo contenga tutto insieme. Può avvenire ad esempio, che TEA generi un messaggio di errore ME! mentre si sta introducendo in esso il programma sorgente. Oppure, anche se in memoria vi è sufficientemente spazio per il programma sorgente, può accadere che TEA generi un messaggio di errore ME! nel corso del processo di assemblaggio. È necessario, in questo caso, salvare sul dispositivo perforatore (IOP) di I/O tutto il testo presente in memoria. Valendosi del comando P (PUNCH) e specificando due numeri di linea, si può spezzare in blocchi il programma sorgente senza nessuna difficoltà. Una volta avvenuta la perforazione dell'intero text buffer, questo può essere tranquillamente cancellato dalla memoria. È possibile, allora passare alla edizione del resto del programma (o di gran parte di esso) nonchè alla sua suddivisione in blocchi man mano che si procede il suo salvataggio sul dispositivo IOP. Al termine dell'ultimo blocco non si dimentichi il segno del dollaro ($).

(2) Avvenuta la completa edizione dei programmi sorgente e il suo salvataggio in blocchi sul dispositivo IOP, si deve introdurre il comando X (ASSEMBLE), al quale TEA risponde con uno dei due termini OCTAL o HEXADECIMAL, a seconda del sistema di numerazione da adottarsi per il listing della seconda passata. Volendo effettuare la conversione dall'uno all'altro dei due sistemi di numerazione, è sufficiente introdurre un CTRL/C, al quale TEA risponderà con C=, essendo ritornato nel modo di comando. Per convertire il sistema di numerazione utilizzato si può allora introdurre una O (OCTAL) oppure una H (HEXADECIMAL). Quando, al termine di questa frase, si introduce la X (ASSEMBLE), TEA risponde specificando il sistema di numerazione che sarà utilizzato nel listing della terza passata e formula la domanda "M OR T?", alla quale si risponde con T, in quanto si desidera assemblare il programma sorgente da banda. Alla successiva

domanda AUTO-LOAD? è possibile rispondere sia con N (NO) che con Y (YES), a seconda o meno che la versione oggetto del programma debba essere caricata in memoria nel corso della seconda passata dell'assembler.

(3) TEA stamperà quindi "R?" per sollecitare all'utente l'inizio del colloquio, dandogli con ciò l'opportunità di posizionare nel lettore del nastro il leader del nastro presente in testa al primo blocco del programma sorgente. Esauriti questi preliminari, si prema il tasto R della telescrivente o del CRT (IONOEC), facendo sì che TEA effettui la lettura dei blocchi del programma sorgente. Al procedere della lettura della banda, TEA forma la tabella dei simboli degli indirizzi. Allorchè legge il segno del dollaro posto a chiusura dell'ultimo blocco, TEA provvede a stampare il numero di errori (REDEF) rivelatisi nel corso della prima passata. In caso di errori il numero di blocco stampato nel messaggio di errore ha la funzione di aiutare a localizzare il blocco di codice sorgente contenente l'errore REDEF. Dopo che, grazie all'impiego dell'editor il codice è stato corretto, si può ridare il via al procedimento di assemblaggio ripartendo dal Passo 2.

Nel caso che non sia stato rivelato alcun errore, si può effettuare la seconda passata del procedimento di assemblaggio. L'estremità iniziale della banda deve quindi essere posta nel lettore di banda ed il dispositivo perforatore IOP di I/O deve essere abilitato nel caso che si desideri una copia del codice oggetto.

Una volta che il dispositivo IOP è stato abilitato (ossia la perforatrice di banda posta in funzione o il registratore a cassette predisposto nel modo RECORD), si deve premere il tasto R della telescrivente o del CRT in risposta alla domanda "R?" di TEA. Premuto il tasto R, TEA comincia a leggere una linea di testo per volta, assemblandola e ponendo in uscita verso il dispositivo IOP il corrispondente byte di 8 bit. Al rilevamento del segno del dollaro posto a chiusura della banda, TEA perfora sulla banda una "coda", stampa il numero degli errori rilevati nel corso della seconda passata e avanza la domanda "R?". Gli errori eventualmente rilevati devono essere corretti valendosi dell'editor ed il procedimento di assemblaggio riavviato da capo (lettura in TEA del nastro perforato in modo da poter formare la tabella dei simboli e degli indirizzi ecc.)

Se, al termine della seconda passata, non è stato rilevato alcun errore, l'estremità iniziale della banda deve essere posizionata un'altra volta nel lettore di banda, premendo quindi il tasto R della telescrivente o del CRT. TEA procede a questo punto alla lettura di una linea di testo, assemblandola e stampando l'indirizzo di memoria ed il suo codice simbolico sul dispositivo di I/O di uscita (IOO), unitamente alla linea di programma sorgente. Allorchè TEA giunge al termine di un blocco del programma sorgente, esso provvede a stampare tanti ritorni di carrello e avanzamenti di linea quanti ne occorrono per arrivare all'inizio della pagina successiva, incrementando il numero di blocco e aprendo una nuova pagina per il nuovo blocco di programma sorgente. Allorchè TEA incontra il segno del dollaro, posto a chiusura della banda, provvede

ancora a stampare tanti ritorni del carrello e avanzamenti di linea quanti sono necessari ad arrivare alla pagina seguente.

Si effettua quindi l'ordinamento dei contenuti della tabella dei simboli degli indirizzi, seguito dalla stampa del risultato di questa operazione. Per programmi di lunghezza considerevole, il tempo occorrente a TEA per l'ordinamento degli elementi della tabella dei simboli degli indirizzi può risultare particolarmente lungo: per ordinare 540 indirizzi simbolici TEA impiega 90 secondi. Una volta che è stata stampata la tabella dei simboli degli indirizzi, TEA provvede a stampare il numero degli errori rilevati nel corso della terza passata.

(4) A questo punto l'utente ha la facoltà di caricare in memoria il programma oggetto mediante il comando B (BOOTSTRAP), oppure di eseguire il programma oggetto precedentemente caricato in memoria per mezzo della funzione di auto-caricamento. Questo è sempre possibile a patto che nel corso della seconda passata non si siano manifestati errori OL (OVERLAP). Se, nel corsò del procedimento di assemblaggio, si è manifestato qualche errore, il blocco (o i blocchi) del programma sorgente che contiene gli errori deve essere riletto in TEA servendosi del comando R (READER), seguito da uno dei comandi A (APPEND), I (INSERT) e C (CHANGE). Ogni blocco di programma sorgente, al termine della sua edizione, deve essere messo da parte sul dispositivo perforatore (IOP) di I/O. Nel caso che si utilizzi una banda perforata, questo nuovo blocco deve essere "giuntato" dentro alla banda di origine *nella sequenza appropriata*. Tale operazione di montaggio costituisce un fastidio che pur bisogna affrontare. È infatti molto più facile lavorare con un programma consistente in dieci blocchi diversi congiunti insieme che non con dieci bande distinte, poichè con queste ultime, è quanto mai difficile evitare che esse siano lette nel microcalcolatore nell'ordine sbagliato.

Come si può notare, il grande vantaggio offerto da questo sistema di assemblare un programma da banda è quello di poter trattare programmi anche molto lunghi, pur disponendo soltanto di 8 o 12 K di memoria R/W, quanto basta cioè per memorizzare TEA insieme ad un pò di testo. Per contro questo modo di assemblare un programma comporta un lavoro decisamente più gravoso che non operando direttamente dalla memoria.

L'Esempio 2-36 illustra le modalità secondo cui avviene la creazione di un programma, la sua suddivisione in vari blocchi ed il suo assemblaggio. È dapprima elencato (L↓) il programma presente in memoria. Si osservi come l'ultima linea di testo contenga un segno di dollaro. Per suddividere il programma in blocchi si è poi utilizzato il comando punch. Il primo blocco (P1-3↓) contiene l'indirizzo definito congiuntamente ad un indirizzo simbolico definito ed al simbolo di istruzione JMP. Il secondo blocco (P4-6↓) contiene l'indirizzo simbolico relativo all'istruzione JMP insieme ad uno zero (il terzo byte dell'istruzione stessa) ed al segno del dollaro. Il risultato elencato nell'Esempio 2-36 è stato ottenuto servendosi di una telescrivente che comprendeva una perforatrice di banda. Sulla stampante sono, perciò, stati stampati anche i risultati

Esempio 2-36. Assemblaggio di un programma da banda servendosi del dispositivo lettore I/O (IOR).

```
C=L
0001
0002            *010 000
0003  START,    JMP
0004            START
0005            0
0006  $
0007
C=P1-3

*010 000
START,JMP

C=P4-6
START
0
$

C=X
OCTAL
M OR T ? T
AUTO-LOAD ?N
R?R
ERRORS DETECTED = 000

R?R00
ERRORS DETECTED = 000
R?R
------

      TYCHON EDITOR-ASSEMBLER V-3                    PAGE 01-001


                        *010 000
010 000 303  START,     JMP


------

      TYCHON EDITOR-ASSEMBLER V-3                    PAGE 02-001

010 001 000             START
010 002 010             0

------

      TYCHON EDITOR-ASSEMBLER V-3                    PAGE 02-002

START  =010 000
ERRORS DETECTED = 000


------


C=
```

"compressi" ottenuti in base al comando punch.

Dopo che il programma è stato trasferito a blocchi su banda perforata, si introduce il comando X (ASSEMBLE). Il listing relativo alla terza passata dovrà risultare in ottale, mentre TEA riceve l'istruzione che il programma da assemblare è contenuto su banda (T). Non si desidera inoltre che, nel corso della seconda passata, il programma oggetto sia caricato in memoria. TEA formula la domanda "R?", per cui si procede a montare il nastro perforato sul lettore di nastro, premendo quindi il tasto R della telescrivente. Avviene così la lettura di entrambi i blocchi della banda, che consente a TEA di formare la tabella dei simboli degli indirizzi. Allorchè legge il segno del dollaro che chiude il secondo blocco, TEA stampa il messaggio di errore e, quindi, ripropone una seconda volta la domanda (R?) di sollecito lettura.

L'estremità iniziale della banda è quindi nuovamente ricollegata nel lettore ed il tasto R della telescrivente premuto. TEA perfora sulla telescrivente la banda con il programma oggetto (si badi ai due caratteri @), stampa il numero di errori rilevati e quindi si pone in attesa che l'utente monti la banda e prema il tasto R ancora una volta, dando poi inizio al listing relativo alla terza passata. È così prodotta la prima pagina del primo blocco (pagina 01-001). Dato che l'indirizzo definito, l'indirizzo simbolico definito ed il simbolo di istruzione JMP sono gli "unici" caratteri contenuti nel primo blocco, TEA stampa la linea che contiene il simbolo JMP e quindi fa avanzare il foglio sino al principio della pagina successiva. Per economia di spazio, le linee vuote tra una pagina e l'altra sono state qui eliminate.

TEA stampa quindi la pagina uno del blocco due (02-001), contenente l'indirizzo simbolico e lo zero relativi all'istruzione JMP. Dette stringhe sono elencate insieme alle dovute informazioni relative agli indirizzi ed ai dati. Avendo rilevato il segno di dollaro con cui termina il secondo blocco, TEA fa avanzare nuovamente il foglio sino al principio di pagina 02-002 dove sono stampati gli elementi ordinati della tabella dei simboli degli indirizzi. È evidente che, essendovi nella tabella un unico indirizzo simbolico, non è possibile accorgersi che vi è stata un'operazione di ordinamento. Dopo la stampa dell'indirizzo simbolico START, TEA provvede a stampare il numero di errori rilevati nel corso della terza passata. Il foglio è poi fatto avanzare sino a che la pagina 02-002 possa ritenersi completata e TEA ritorna al modo di comando.

CARICAMENTO IN MEMORIA DI UN PROGRAMMA ·OGGETTO

Nel caso che si decida di non valersi della funzione di autocaricamento di TEA per caricare in memoria la versione oggetto del programma nel corso della seconda passata dell'assembler, bisogna valersi del comando B (BOOTSTRAP). L'impiego di questo comando permette di caricare in *qualunque* momento in memoria *qualsiasi* programma oggetto generato da TEA.

1. Per caricare in memoria un programma oggetto, occorre collocare

nel lettore di nastro il leader (parte iniziale) della banda perforata (Appendice A).

2. Con TEA nel modo di comando, introduci il comando B (BOOTSTRAP).
3. Sposta la levetta dello switch di selezione della telescrivente nella posizione START.
4. Allorchè la lettura del nastro nel microcalcolatore è terminata, TEA esegue un confronto mediante la procedura "comparesum" (analoga a checksum) per controllare che non vi siano stati errori di lettura da banda dei codici ivi presenti. Nel caso che vi sia stato qualche errore, TEA provvede a stampare il messaggio di errore BAD CHECKSUM (somma di controllo errata). In questo caso, è bene non fare nessun tentativo di eseguire ugualmente il programma appena letto in memoria. Poichè non vi è nessuna possibilità di stabilire dove sia avvenuto l'errore, occorre rileggere in memoria una seconda volta l'intera banda. se gli errori persistono, è probabile che sia la banda a contenere un errore, rendendo perciò necessaria la ripetizione dell'assemblaggio del programma.

Nel caso che TEA ritorni senz'altro al modo di comando al termine della lettura in memoria dell'intera banda perforata, si può essere certi che non vi sono stati errori. Può quindi essere introdotto il comando Z, mediante il quale il controllo è trasferito ad un monitor di sistema o ad un debugger. Per dare inizio all'effettiva esecuzione del programma ci si può quindi servire di uno di questi programmi.

RIFERIMENTI

1. Titus, C.A. e Titus J.A., DBUG: An 8080 *Interpretive Debugger*, Howard W. Sams & Co., Inc., Indianapolis, IN 1977.

BIBLIOGRAFIA

1. Titus, C.A., Rony, P.R., Larsen, D.G., e Titus, J.A. *8080/8085 Software Design, Book 1*, Howard W. Sams & Co., Inc., Indianapolis, IN 1978.

2. Titus, C.A., *8080/8085 Software Design, Book 2*, Howard W. Sams & Co., Inc., Indianapolis, IN 1979.

CAPITOLO 3

Convenzioni relative ai dispositivi di I/O

Per poter funzionare, TEA deve avere la possibilità di accedere a dispositivi di I/O, le cui funzioni possono variare a seconda dei casi: introdurre i comandi o linee di testo, listare il contenuto del text buffer e salvare, memorizzandolo in qualche dispositivo periferico, il contenuto del buffer o un programma oggetto. Le subroutine di I/O contenute in TEA, adibite a questo scopo, sono in tutto sei. Tuttavia, non sono chiamate direttamente da TEA che vi accede passando attraverso una "tabella di salto" (jump table). È detta tabella di salto una normale tabella che contiene diverse istruzioni di salto a tre byte. Anzichè effettuare direttamente la chiamata di una delle subroutine di I/O, TEA chiama una delle locazioni di memoria contenente una di queste istruzioni di salto. Con l'esecuzione dell'istruzione di salto il controllo del programma è restituito all'interno di TEA alla subroutine di I/O.

L'utente può facilmente sostituire i dispositivi di I/O ai quali TEA ha accesso, cambiando il secondo e terzo byte delle istruzioni di salto in tabella. Il cambiamento del secondo e terzo byte delle istruzioni di salto, consente, inoltre, l'utilizzazione da parte di TEA di una subroutine predisposta dall'utente. Volendo scrivere subroutine di I/O funzionanti correttamente nell'ambito di TEA, è comunque necessario che l'utente tenga ben presente le caratteristiche delle subroutine di I/O già comprese in TEA. Note tali caratteristiche, si possono scrivere nuove subroutine di caratteristiche analoghe, anche se il loro impiego riguarda l'accesso a dispositivi periferici che sono differenti.

CARATTERISTICHE DELLE SUBROUTINE DI I/O

Le subroutine di I/O comprese in TEA sono sei. Questo capitolo si occupa, per l'appunto, delle modalità di funzionamento di ciascuna di

esse e dei registri interessati. Nel caso che nuove subroutine di I/O debbano servirsi di registri diversi da quelli indicati, è indispensabile che questi registri *siano salvati nello stack prima che possano essere impiegati nella subroutine, e che, in seguito, siano ripristinati prima del ritorno dell'8080 dalla subroutine.*

IOTEST

Questa subroutine effettua *una volta per tutte* un test del flag di dati-disponibili della tastiera. Al ritorno dell'8080 dalla subroutine IOTEST, il registro A conterrà zero oppure avrà un contenuto diverso da zero a seconda che non sia stato premuto nessun tasto oppure sia stato premuto un qualsiasi tasto. Il test del flag di zero può essere effettuato subito dopo la restituzione del controllo da parte della subroutine. La funzione di quest'ultima non è quella di porre in ingresso il codice ASCII a sette-otto bit proveniente dalla tastiera nel caso di tasto premuto, ma unicamente di effettuare il test del flag della tastiera. IOTEST è per l'appunto, la prima subroutine il cui indirizzo è contenuto nella tabella di salto. La sua chiamata fa sì che l'8080 salti alla subroutine INCHK contenuta in TEA e riportata nell'Esempio 3-1

Esempio 3-1. La subroutine INCHK (keyboard flag chek: controllo del flag della tastiera).

```
006 121 333  INCHK,  IN      /JUST GET THE KEYBOARD FLAG
006 122 001          001
006 123 346          ANI     /AND SET THE FLAGS BASED ON ITS STATE
006 124 001          001
006 125 311          RET
```

Il compito della subroutine INCHK si riduce a cambiare il contenuto del registro A e lo stato dei flag. Dovendo utilizzare dei registri addizionali in una versione della subroutine INCHK, occorre che essi siano posti sullo stack all'inizio della subroutine e prelevati al termine della stessa. Con l'esecuzione dell'istruzione RET con cui termina INCHK, l'8080 fa ritorno all'istruzione presente in memoria dopo la chiamata CALL di IOTEST.

IONOEC

Il compito di questa subroutine consiste nell'effettuare continui test del flag di dati-disponibili della tastiera. Nel caso che il flag sia in uno logico, l'8080 pone in ingresso il codice ASCII battuto sulla tastiera. Il bit più significativo del codice di otto bit (il bit di parità) è posto quindi a zero. L'8080 fa ritorno dalla subroutine IONOEC con il codice ASCII di sette bit contenuto nel registro A. Il relativo carattere ASCII *non* è però stampato sulla stampante. IONOEC costituisce la seconda subroutine con indirizzo chiamato da un'istruzione di salto della tabella di salto. Chiamando la subroutine TTYI contenuta in TEA e riportata nell'Esempio 3-2.

Esempio 3-2. La subroutine TTYI (keyboard input, no echo: ingresso da tastiera senza eco).

```
006 152 333  TTYI,   IN      /GET THE PRINTER'S AND KEYBOARD'S
006 153 001          001     /FLAGS
006 154 346          ANI     /SAVE JUST THE KEYBOARD'S FLAG
006 155 001          001
006 156 312          JZ      /IF THE FLAG IS ZERO, WAIT FOR IT
006 157 152          TTYI    /TO BE A LOGIC ONE, MEANING THAT A
006 160 006          0       /KEY HAS BEEN PRESSED.
006 161 333          IN      /A KEY WAS PRESSED, INPUT ITS
006 162 000          000     /ASCII VALUE.
006 163 346          ANI     /MASK OUT ANY PARITY BIT
006 164 177          177
006 165 311          RET     /AND RETURN WITH IT IN THE A REGISTER
```

La subroutine TTYI si limita a cambiare il contenuto del registro A e lo stato dei flag. Nel caso che una nuova versione della subroutine TTYI abbia bisogno di impiegare dei registri addizionali, questi devono essere aggiunti allo stack all'inizio della subroutine e prelevati da esso al termine della medesima. In seguito all'esecuzione dell'istruzione RET con la quale termina la subroutine TTYI, l'8080 fa ritorno all'istruzione presente in memoria subito di seguito alla chiamata CALL di IONOEC.

IOECHO

La funzione di questa subroutine è quella di permettere l'ingresso di un carattere da una tastiera ASCII e la sua stampa sulla stampante di una telescrivente o lo schermo di un CRT. Avvenuta la stampa del carattere, si procede ad un nuovo controllo della tastiera al fine di stabilire se, mentre la stampante era in azione, non sia stato introdotto un CTRL/C. In caso affermativo, lo stack pointer è resettato e l'8080 salta alle istruzioni corrispondenti all'inizio di TEA, in base alle quali TEA passa al modo di comando e stampa C=. Nel caso che sia stato premuto un tasto non corrispondente ad un CTRL/C, la circostanza è ignorata e l'8080 ritorna alla sezione di TEA in cui è avvenuta la chiamata della subroutine IOECHO. Quest'ultima costituisce la terza subroutine alla quale si indirizza un'istruzione di salto nella tabella di salto. Con la chiamata della subroutine IOECHO l'8080 effettua un salto indietro alla subroutine TTYIN contenuta in TEA e riportata nell'Esempio 3-3.

La subroutine TTYIN cambia il contenuto dei registri A e B e gli stati dei flag. Nel caso che una nuova versione di TTYIN preveda l'impiego di altri registri, è necessario che questi ultimi siano aggiunti allo stack al principio del subroutine e prelevati da esso al termine della medesima. Con l'esecuzione delle istruzioni RZ o RNZ con le quali termina TTYIN (nell'ipotesi che non sia stato introdotto un CTRL/C) l'8080 ritorna ad eseguire l'istruzione che si trova in memoria subito dopo la chiamata CALL di IOECHO.

A proposito di questa subroutine, è opportuno procedere con la massima cautela, soprattutto se i registri sono aggiunti allo stack e le istruzioni di ritorno condizionato sono lasciate nella subroutine. Si

consiglia di trasferire queste ultime in istruzioni di salto condizionato allo scopo di facilitare lo svuotamento dello stack.

IOP

La funzione di questa subroutine è quella di aiutare a perforare un carattere su banda oppure a registrarlo su di un'audio-cassetta. Chiamando la subroutine IOP, l'8080 effettua un salto alla subroutine TTYOUT, riportata poco sopra, mediante un'istruzione di salto presente nella tabella di salto. Come si può notare nell'Esempio 3-3, la subroutine TTYOUT costituisce in realtà parte integrante della subroutine TTYIN. Il motivo per cui essa è chiamata risiede nel fatto che nella maggioranza dei casi al meccanismo della stampante è collegata meccanicamente una perforatrice di banda di una telescrivente. Questo comporta che, con la perforatrice in azione, tutti i caratteri posti in uscita sulla stampante risultano anche perforati su banda. Analogamente, ogni carattere posto in uscita sulla perforatrice sarà anche stampato sulla stampante.

La subroutine IOP deve essere chiamata solamente quando il valore da perforare si trova già nel registro A. La subroutine TTYOUT provvede a mettere provvisoriamente da parte nel registro B tale valore. Con il ritorno di TEA dalla subroutine TTYOUT, il valore si troverà ancora in entrambi i registri A e B. La subroutine, inoltre, modifica i flag. La

Esempio 3-3. La subroutine TTYIN (poni in ingresso un carattere, stampalo e verifica se c'è un CTRL/C).

```
006 166 315  TTYIN,  CALL     /GET A KEYBOARD CHARACTER
006 167 152          TTYI     /AND PRINT IT.
006 170 006          0
006 171 107  TTYOUT, MOVBA    /SAVE THE CHARACTER IN B
006 172 333          IN
006 173 001          001
006 174 346          ANI
006 175 004          004      /CHECK TO SEE IF THE PRINTER
006 176 312          JZ       /IS READY.  IT ISN'T, KEEP CHECKING
006 177 172          TTYOUT+1
006 200 006          0
006 201 170          MOVAB    /PRINTER IS READY, GET THE CHARACTER
006 202 323          OUT
006 203 000          000      /AND PRINT IT
006 204 315          CALL     /SEE IF ANY KEYS HAVE BEEN PRESSED
006 205 000          IOTEST
006 206 030          0
006 207 170          MOVAB    /GET THE PRINTER CHARACTER BACK IN A
006 210 310          RZ       /NO KEYS PRESSED
006 211 315          CALL     /A KEY WAS PRESSED, FIND OUT WHAT IT IS
006 212 003          IONOEC
006 213 030          0
006 214 376          CPI
006 215 003          CTRLC    /WAS A CTRL-C TYPED IN?
006 216 170          MOVAB    /GET THE OLD PRINTER CHARACTER
006 217 300          RNZ      /NO, NOT A CNTRL-C, SO IGNORE IT
006 220 061          LXISP    /IT WAS A CNTRL-C,
006 221 240          STACK    /RESET THE STACK POINTER
006 222 030          0
006 223 303          JMP      /AND SEE WHAT THE USER'S UP TO
006 224 153          QUEST
006 225 000          0
```

chiamata della subroutine IOP avviene nel caso che si impieghi il comando P (PUNCH: perfora) dell'editor, nonchè nel corso della seconda passata dell'assembler, quando si desidera ottenere in uscita una versione oggetto del programma sorgente.

La subroutine TTYOUT si avvale dei registri A e B e modifica altresì lo stato dei flag. Nel caso di impiego di altri registri in una nuova versione di TTYOUT, questi dovranno essere aggiunti allo stack al principio della subroutine e prelevati da esso al termine della medesima. All'esecuzione dell'istruzione RET, con la quale si conclude la subroutine TTYOUT, l'8080 ritorna all'istruzione presente in memoria di seguito alla chiamata CALL di IOP.

IOR

La funzione di questa subroutine è quella di permettere la lettura di un singolo valore da un lettore di banda o da un registratore a cassette. Poichè il lettore di banda perforata e la tastiera della generalità delle telescriventi sono collegati meccanicamente in parallelo, questa subroutine si presenta in forma quasi uguale alla subroutine IONOEC (TTYI). L'unica differenza sta nel fatto che in primo luogo si ha l'esecuzione di un'istruzione OUT in modo che il relè di controllo lettura della telescrivente riceva un impulso per l'avanzamento della banda. La subroutine incontrata nella tabella di salto. In seguito alla sua chiamata da parte di

Esempio 3-4. La subroutine IORDR (fornisci un impulso al relè di controllo della lettura, poni in ingresso un carattere).

```
006 126 323   IORDR,  OUT     /PULSE THE READER CONTROL RELAY
006 127 001           001
006 130 000           0       /THERE IS ENOUGH ROOM HERE FOR ANY
006 131 000           0       /TIME DELAY SOFTWARE REQUIRED FOR A
006 132 000           0       /SOFTWARE CONTROLLED READER-CONTROL
006 133 000           0       /RELAY.
006 134 000           0
006 135 000           0
006 136 000           0
006 137 000           0
006 140 000           0
006 141 000           0
006 142 000           0
006 143 000           0
006 144 000           0
006 145 000           0
006 146 000           0
006 147 000           0
006 150 000           0
006 151 000           0
006 152 333   TTYI,   IN      /GET THE PRINTER'S AND KEYBOARD'S
006 153 001           001     /FLAGS
006 154 346           ANI     /SAVE JUST THE KEYBOARD'S FLAG
006 155 001           001
006 156 312           JZ      /IF THE FLAG IS ZERO, WAIT FOR IT
006 157 152           TTYI    /TO BE A LOGIC ONE, MEANING THAT A
006 160 006           0       /KEY HAS BEEN PRESSED.
006 161 333           IN      /A KEY WAS PRESSED, INPUT ITS
006 162 000           000     /ASCII VALUE.
006 163 346           ANI     /MASK OUT ANY PARITY BIT
006 164 177           177
006 165 311           RET     /AND RETURN WITH IT IN THE A REGISTER
```

TEA, l'istruzione della tabella di salto trasferisce il controllo del programma alla subroutine IORDR contenuta in TEA. Questa subroutine è riportata nell'Esempio 3-4.

Sia l'editor che l'assembler facenti parte di TEA si valgono del dispositivo di IOR e della relativa subroutine. Con l'introduzione del comando R (READER), si impiega il dispositivo IOR per porre in ingresso al text buffer un programma sorgente mediante uno dei comandi A (APPEND), I (INSERT) o C (CHANGE). L'Assembler si varrà del dispositivo IOR nel caso che il programma sorgente da assemblare sia presente su banda ("M OR T?" T). Questo dispositivo *deve essere in grado* di leggere una linea di testo per volta. È proprio questo il motivo per cui è necessario il relè di controllo della lettura della telescrivente, senza il quale l'8080 è privato di qualsiasi mezzo di controllo sul lettore di banda. Ne consegue che risulterà estremamente difficile, se non addirittura impossibile, impiegare TEA con una telescrivente priva di un relè di controllo della lettura, oppure con un registratore a cassette poichè la maggior parte di essi non si presta a leggere una linea di testo per volta.

La subroutine IOR (IORDR) modifica il contenuto del registro A e lo stato dei flag. Nel caso che TEA acceda ad una nuova subroutine, e si renda necessario il ricorso a dei registri addizionali, questi ultimi devono essere aggiunti allo stack al principio della subroutine IORDR e prelevati da esso al termine della medesima. Con l'esecuzione dell'istruzione RET, con la quale termina la subroutine IORDR, l'8080 ritorna all'istruzione memorizzata di seguito alla chiamata CALL di IOR.

IOO

La chiamata di questa subroutine permette che il contenuto del registro A sia posto in uscita sulla stampante della telescrivente o sulla logica/memoria di controllo dello schermo del CRT. Questa è, altresì, la quinta ed ultima subroutine alla quale si accede tramite la tabella di salto. Allorchè TEA effettua la chiamata della subroutine IOO la tabella di salto fa avvenire il trasferimento del controllo alla subroutine TTYOUT riportata nella sezione IOECHO di questo capitolo. La subroutine TTYOUT si serve dei registri A e B e modifica inoltre lo stato dei flag. Nel caso che, per una nuova versione della subroutine IOO, si renda necessario l'impiego di registri addizionali, occorre servirsi dello stack come memoria provvisoria per il contenuto dei registri che la subroutine utilizzerà. Con l'esecuzione dell'istruzione RET al termine di TTYOUT, TEA ritornerà all'istruzione memorizzata subito dopo la chiamata CALL di IOO.

La maggior parte dei dispositivi periferici, purtroppo, necessiterà di subroutine di I/O più lunghe e complesse di quelle già comprese in TEA, che sono, invece, corte e compatte per la semplice ragione che una tastiera di telescrivente ed un lettore di banda consistono in pratica nello stesso dispositivo, esattamente come la stampante e la perforazione di banda di una telescrivente.

In base al listing delle subroutine di I/O contenute internamente a TEA, è facile osservare come debbano essere configurate le eventuali subroutine di I/O differenti, quali siano i registri che si possono utilizzare e quale debba essere il loro contenuto al momento della chiamata della subroutine e a quello del ritorno da essa. Tutto questo dovrebbe facilitare grandemente la compilazione di nuove subroutine di I/O.

MODIFICA DELLA TABELLA DI SALTO DI I/O

Negli Esempi 3-5 (ottale) e 3-6 (esadecimale) è riportata la tabella di salto che TEA trascrive in memoria R/W subito dopo essere stato avviato. L'impiego di questa tabella di salto è fondamentale allorchè l'unico dispositivo di I/O consiste in una telescrivente dotata di un lettore/perforatrice di banda.

Dovendo eseguire una delle subroutine di I/O, TEA chiama una delle istruzioni memorizzate agli indirizzi IOTEST, IONOEC, IOECHO, IOP, IOR o IOO. Allorchè giunge ad una di queste sei "subroutine", l'8080 salta immediatamente a delle subroutine contenute all'interno di

Esempio 3-5. La tabella di salto generata e utilizzata da TEA (ottale).

```
                        *030 000
030 000 303   IOTEST,   JMP      /GET THE INPUT DEVICE
030 001 121             INCHK    /DATA READY FLAG
030 002 006             0
030 003 303   IONOEC,   JMP      /INPUT A TTY OR CRT CHARACTER
030 004 152             TTYI     /INTO A & B, BUT NO ECHO !
030 005 006             0
030 006 303   IOECHO,   JMP      /INPUT A CHARACTER INTO A & B
030 007 166             TTYIN    /BUT ECHO IT ON THE PRINTER.
030 010 006             0
030 011 303   IOP,      JMP      /PUNCH OUT THE CHARACTER IN A
030 012 171             TTYOUT   /THIS COULD BE A TTY PUNCH OR
030 013 006             0        /AUDIO CASSETTE
030 014 303   IOR,      JMP      /READ A CHARACTER INTO A&B
030 015 126             IORDR    /FROM A TTY READER OR AUDIO CASSETTE
030 016 006             0
030 017 303   IOO,      JMP      /PRINT THE CHARACTER CONTAINED IN A.
030 020 171             TTYOUT   /THIS COULD BE A CRT, TTY OR CASSETTE.
030 021 006             0
```

Esempio 3-6. La tabella di salto generata e utilizzata da TEA (esadecimale).

```
                        *030 000
18 00 C3      IOTEST,   JMP      /GET THE INPUT DEVICE
18 01 51                INCHK    /DATA READY FLAG
18 02 06                0
18 03 C3      IONOEC,   JMP      /INPUT A TTY OR CRT CHARACTER
18 04 6A                TTYI     /INTO A & B, BUT NO ECHO !
18 05 06                0
18 06 C3      IOECHO,   JMP      /INPUT A CHARACTER INTO A & B
18 07 76                TTYIN    /BUT ECHO IT ON THE PRINTER.
18 08 06                0
18 09 C3      IOP,      JMP      /PUNCH OUT THE CHARACTER IN A
18 0A 79                TTYOUT   /THIS COULD BE A TTY PUNCH OR
18 0B 06                0        /AUDIO CASSETTE
18 0C C3      IOR,      JMP      /READ A CHARACTER INTO A&B
18 0D 56                IORDR    /FROM A TTY READER OR AUDIO CASSETTE
18 0E 06                0
18 0F C3      IOO,      JMP      /PRINT THE CHARACTER CONTAINED IN A.
18 10 79                TTYOUT   /THIS COULD BE A CRT, TTY OR CASSETTE.
18 11 06                0
```

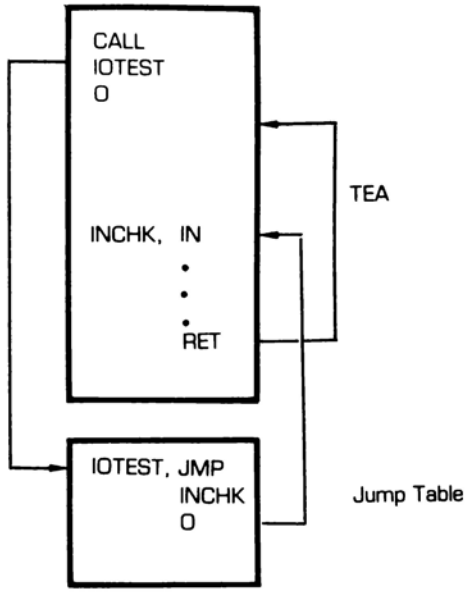


Fig. 3-1. Procedura di, accesso alla subroutine IOTEST (INCHK), contenuta in TEA, con il tramite della tabella di salto.

TEA. Effettuata l'operazione di I/O opportuna, si ha l'esecuzione dell'istruzione RET (oppure di un'istruzione di ritorno condizionato), per cui il controllo del programma ritorna all'istruzione memorizzata subito dopo la chiamata di IOTEST, IONOEC, IOECHO, IOP, IOR oIOO (Fig. 3-1).

La tabella di salto indirizza TEA alla subroutine di I/O appropriate. Sostituendo indirizzi diversi a quelli delle istruzioni di salto della tabella, si può far sì che TEA acceda a nuove subroutine di I/O ed a nuovi dispositivi di I/O interfacciati a cura dell'utente con il suo microcomputer basato sull'8080, 8085 o Z80. (Fig. 3-2)

La conoscenza dei principi su cui si basa la tabella di salto e dei registri che possono o meno essere utilizzati è di grande importanza. Anche se può rendersi necessario l'impiego di indirizzi di dispositivo diversi da quelli, 000 e 001, utilizzati nelle subroutine di I/O anche se le posizioni dei bit della parola di stato (status word) riservate come indicatori dello stato dei flag di dati disponibili e di trasmettitore pronto possono variare, i criteri di trasferimento dei dati e di controllo mediante flag sono sempre gli stessi.

Si immagini di volere porre in ingresso a TEA da un lettore di banda ultra-veloce un programma sorgente che deve poi essere assemblato. Nel corso della seconda passata dell'assembler, il programma oggetto deve essere posto in uscita verso un registratore a cassette e, nel corso della terza passata, il listing del programma assemblato deve essere stampato su di una stampante di telescrivente. Si può allora utilizzare la tabella di salto riportata nell'Esempio 3-7.

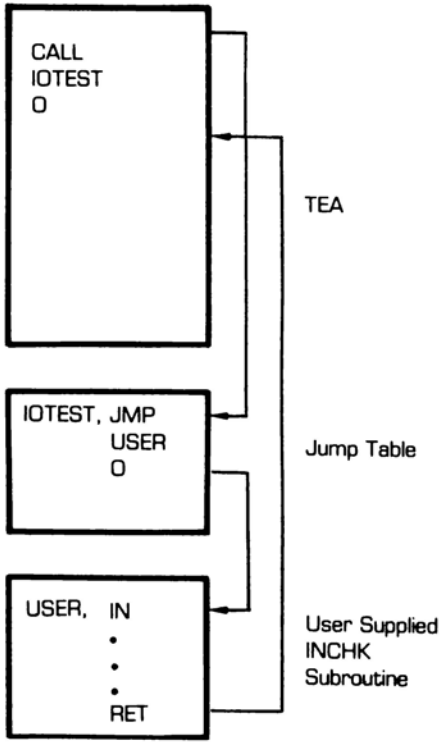


Fig. 3-2. Modalità di accesso ad una subroutine IOTEST (INCHK) predisposta dall'utente.

L'esame dell'Esempio 3-7 dovrebbe mettere bene in evidenza come sia stato necessario cambiare il secondo e terzo byte delle istruzioni di salto relative ai dispositivi IOP e IOR. Il dispositivo IOP è, in questo caso, un registratore a cassette anzichè una perforatrice di banda di una telescri-

Esempio 3-7. Modifica della tabella di salto per poterla utilizzare con un registratore a cassette ed un lettore di nastro ad elevata velocità.

```
                        *030 000
030 000 303  IOTEST,    JMP      /GET THE INPUT DEVICE
030 001 121             INCHK    /DATA READY FLAG
030 002 006             0
030 003 303  IONOEC,    JMP      /INPUT A TTY OR CRT CHARACTER
030 004 152             TTYI     /INTO A & B, BUT NO ECHO !
030 005 006             0        .
030 006 303  IOECHO,    JMP      /INPUT A CHARACTER INTO A & B
030 007 166             TTYIN    /BUT ECHO IT ON THE PRINTER.
030 010 006             0
030 011 303  IOP,       JMP      /THE "PUNCH DEVICE" IS NOW
030 012 306             CASSW    /AN AUDIO CASSETTE
030 013 123             0
030 014 303  IOR,       JMP      /THE "READER DEVICE" IS NOW
030 015 256             HSR      /A HIGH SPEED PAPER TAPE
030 016 124             0        /READER
030 017 303  IOO,       JMP      /PRINT THE CHARACTER CONTAINED IN A.
030 020 171             TTYOUT   /THIS COULD BE A CRT, TTY OR CASSETTE.
030 021 006             0
```

vente. L'indirizzo corrispondente al secondo e terzo byte dell'istruzione di salto è stato cambiato da 006 171 (0679 hex) in 123 306 (53C6 hex). L'indirizzo costituito dal secondo e terzo byte dell'istruzione di salto per il dispositivo IOR è stato anch'esso cambiato da 006 126 (0656 hex) a 124 256 (54AE hex), in modo che la lettera del programma sorgente avvenga su un lettore di banda ultraveloce anzichè su quello della telescrivente.

È ovvio che cambiare soltanto il secondo e il terzo byte delle istruzioni di salto non basta: occorre scrivere le nuove subroutine di I/O e quindi memorizzarle. Con le variazioni apportate alla tabella di salto, la nuova subroutine IOP deve essere memorizzata a partire dalla locazione 123 306 (53C6 hex) e la nuova subroutine IOR a partire dalla locazione 124 256 (54AE hex). Queste sono subroutine *appositamente scritte dall'utente* per tali dispositivi di I/O. Esse devono essere conformi alle specifiche sottolineate nelle prime sezioni di questo capitolo.

Si supponga ora di non disporre di una telescrivente e di volere, in alternativa, servirsi di TEA con un CRT e un registratore a cassette. La tabella di salto può essere modificata come risulta dall'Esempio 3-8.

Le subroutine KYCHK, KEYIN e KEYIN2 non sono altro che le subroutine scritte per la tastiera del CRT, sulla falsa riga delle subroutine IOTEST, IONOEC e IOECHO. Gli indirizzi simbolici CASSW e CASSR costituiscono gli indirizzi simbolici corrispondenti rispettivamente alla subroutine di lettura e a quella di scrittura di un carattere da e su cassetta. L'indirizzo simbolico CRTOUT fa riferimento alla subroutine di I/O che stampa "un carattere sullo schermo del CRT. Uno dei compiti principali che si richiedono alla subroutine IOO (o alla sua equivalente) è di non limitarsi a stampare il carattere contenuto nel registro A, *ma altresì di ispezionare la tastiera per scoprire l'eventuale introduzione di un CTRL/C*. Il motivo per cui la tastiera deve essere tenuta in osservazione alla ricerca di un CTRL/C dipende dal fatto che la formazione del listing in base al comando L (LIST) dell'editor e la sua riproduzione da parte dell'assembler in fase di terza passata devono poter essere interrotte in qualsiasi momento. Anche se non è detto che

Esempio 3-8. Modifica della tabella di salto per poterla utilizzare con un registratore a cassette ed un CRT.

```
                        *030 000
030 000 303  IOTEST,    JMP      /CHECK THE CRT'S KEYBOARD
030 001 032             KYCHK
030 002 100             0
030 003 303  IONOEC,    JMP      /WAIT FOR THE KEYBOARD'S FLAG, INPUT
030 004 076             KEYIN    /THE KEY CODE.  DO NOT PRINT THE
030 005 100             0        /CHARACTER
030 006 303  IOECHO,    JMP      /WAIT FOR THE KEYBOARD'S FLAG, INPUT
030 007 107             KEYIN2   /THE KEY CODE AND PRINT THE CHARACTER
030 010 100             0
030 011 303  IOP,       JMP      /THE PUNCH DEVICE IS NOW AN
030 012 306             CASSW    /AUDIO CASSETTE
030 013 123             0
030 014 303  IOR,       JMP      /THE READER DEVICE IS NOW
030 015 374             CASSR    /AN AUDIO CASSETTE
030 016 123             0
030 017 303  IOO,       JMP      /THE OUTPUT DEVICE IS NOW
030 020 166             CRTOUT   /A CRT.
030 021 100             0
```

nella nuova subroutine IOO questi passi siano presenti, è pur vero che la maggior parte delle nuove subroutine IOO li prevede.

Cambiamento degli Indirizzi nella Tabella di Salto

Allorchè TEA è avviato nella fase iniziale, esso comincia con il creare in memoria R/W un duplicato della tabella di salto contenuta al suo interno. Nel caso che sia necessario aggiungere alla tabella di salto gli indirizzi di nuove subroutine di I/O, queste istruzioni non devono essere eseguite.

Per cambiare gli indirizzi della tabella di salto, a cominciare da 030 000 (1800 hex), si può ricorrere ad un monitor di sistema, un debugger, un pannello di controllo o un lettore di banda. Detti indirizzi, tuttavia, devono essere cambiati solamente dopo che TEA è stato caricato in memoria. È evidente che se TEA è contenuto in memoria EPROM, non vi è alcun bisogno di provvedere al caricamento di TEA in memoria. Per apportare eventuali modifiche alla tabella di salto si devono, prima di tutto, scrivere in memoria tutte e sei le istruzioni di salto *originali*, a partire dalla locazione 030 000 (1800 hex). Queste istruzioni sono elencate negli Esempi 3-5 e 3-6.

Trascritte in memoria tutte e sei le istruzioni di salto complete dei rispettivi indirizzi, bisogna cambiare questi indirizzi contenuti nel secondo e terzo byte delle istruzioni di salto in modo che corrispondano agli indirizzi di partenza delle nuove subroutine di I/O. Effettuati questi cambiamenti, occorre far partire TEA dall'indirizzo 000 030 (0018 hex). Facendo partire TEA da questo indirizzo, anzichè da 000 000, esso non trascriverà più in memoria R/W la tabella di salto. Anche nel caso che TEA sia memorizzata in EPROM, può sempre essere cambiata la tabella di salto in quanto essa è anche memorizzata in memoria R/W. È evidente che, affinchè TEA possa funzionare correttamente, occorre che l'indirizzo in memoria R/W sia 030 000 (1800 hex).

LIMITI DEL DISPOSITIVO DI I/O

Esistono una subroutine di lettura (IOR) di uso generale ed una subroutine di perforazione (IOP) di uso generale. I dispositivi più comuni tra quelli utilizzati con queste due subroutine sono un lettore-/perforatrice di banda ed un registratore a cassette nel modo di funzionamento RECORD/PLAY (registra/riproduci). L'impiego classico della subroutine IOR ha luogo nelle operazioni di ingresso di un programma sorgente (testo) eseguite con l'aiuto dell'editor di TEA. Dopo che l'utente lo ha abilitato con il comando R (READER), questo dispositivo può solamente leggere un programma sorgente, ossia del testo. Avvenuta l'edizione del programma sorgente o del testo tramite l'editor, esso può essere riposto sul dispositivo IOP valendosi del comando P (PUNCH). Com'è noto il dispositivo IOP può essere utilizzato per conservare una linea, un gruppo di linee o il contenuto dell'intero text buffer.

L'impiego delle subroutine IOR e IOP interessa anche l'assembler. Il dispositivo IOP è utilizzato nel corso della seconda passata dell'assem-

bler, allorchè si genera la versione oggetto del programma sorgente che è poi posta in uscita verso il dispositivo IOP. Il dispositivo IOR è invece utilizzato dall'assembler quando il programma sorgente può essere contenuto al completo in memoria R/W. Questo fatto comporta che l'assembler effettua la lettura di una linea *del programma sorgente dal dispositivo IOR* provvedendo quindi ad elaborarla. Terminato con una linea del programma sorgente, l'assembler passa a leggere dal dispositivo IOR un'altra linea del programma sorgente, la elabora e via di questo passo.

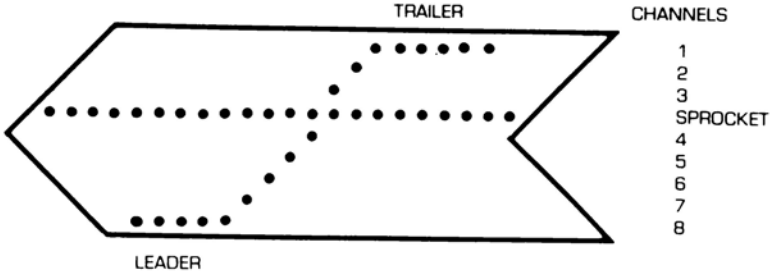
A seconda della particolarità del registratore a cassette o del lettore di banda perforata a disposizione, può avvenire che il funzionamento in questo modo intermittente non sia possibile. Nelle normali telescriventi esiste già un relè di controllo lettura adibito proprio a questo compito. Nel caso che la telescrivente di cui si dispone sia sprovvista di un relè di controllo lettura, la sua aggiunta non comporta problemi nè tecnici nè di spesa.

Qualunque dispositivo che possa essere utilizzato per leggere una banda del programma sorgente per volta si presta senza problemi a porre in ingresso al calcolatore il programma sorgente nel corso del procedimento di assemblaggio. Questa possibilità esiste sempre purchè ci si ricordi di introdurre un "T" di risposta alla domanda "M OR T?", mentre è in corso l'esecuzione dell'assembler (comando X o ASSEMBLE). Purtroppo alla maggior parte dei registratori a cassette manca proprio questa capacità. Dopo aver letto una linea del testo, il registratore dovrebbe fermarsi, aspettando che l'assembler faccia richiesta di un'altra linea di programma sorgente. Esistono, d'altronde, telescriventi, registratore digitale e lettori di banda perforata che sono dotati di questa capacità, ma se il dispositivo IOR di cui si dispone ne è sprovvisto, è gioco forza accontentarsi di utilizzare il modo di assemble M, ossia il modo di memoria.

APPENDICE A

Formato della banda per il programma TEA

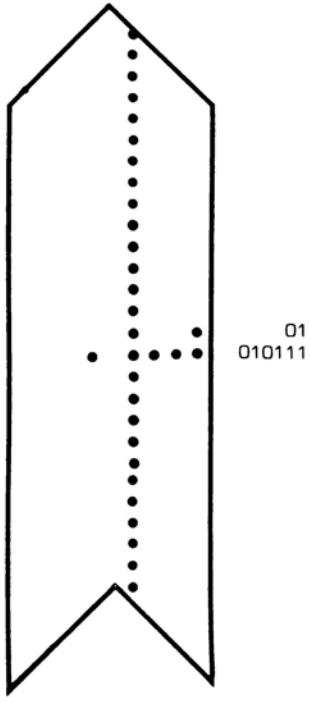
Il formato della banda perforata è il seguente:



Il canale 8 (se è l'unico canale perforato) costituisce il leader/trailer (LDR, TRLR; 200 ottale ovvero 80 esadecimale).

Quando è perforato soltanto il canale 7, le quattro configurazioni che seguono (parole) costituiscono un'informazione relativa ad un indirizzo. Le parole che rappresentano i dati e le informazioni relative agli indirizzi sono spezzate in due byte, il primo dei quali di 2 bit e l'altro di 6 bit. Perciò il numero 127 (57 esadecimale) si presenterebbe nella forma sotto riportata:

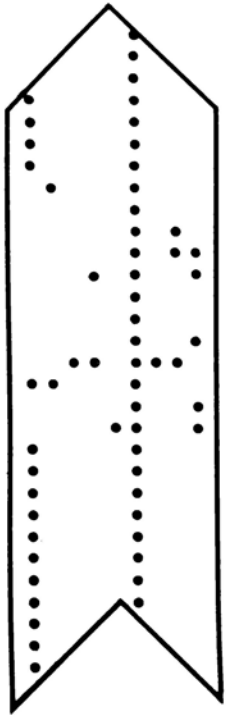
01 010 111 = 01 010111
0101 0111 = 01 010111



Il seguente nastro è perforato sia con codici di indirizzo che di dati. Un indirizzo pari a 002 321 (02 D1 esadecimale) sarebbe:

002 = 00 000 010 = 00 000010
321 = 11 010 001 = 11 010001

02 = 0000 0010 = 00 000010
D1 = 1101 0001 = 11 010001



LEADER(LDR)

ADDRESS FLAG
00 HI ADDRESS
000010
11 LO ADDRESS
010001
00 DATA
000000
01 DATA
110110
COMPRESUM FLAG
01 COMPARESUM
001001

TRAILER (TRLR)

APPENDICE B

Modifica del comando Z

Come già si è avuto occasione di osservare, volendo passare il controllo del programma ad un monitor di sistema o ad un debugger ci si può valere del comando Z. Comunque, dato che il comando Z è presente in una configurazione appartenente a TEA, il microprocessore non fa altro che restituire il controllo del programma all'inizio di TEA in corrispondenza della locazione di memoria 000 030 (0018 hex). All'introduzione del comando M (MEMORY), il controllo del programma è anch'esso trasferito a questo punto di TEA. Ne risulta che, se si desidera trasferire il controllo ad un monitor di sistema o un debugger, occorre cambiare l'indirizzo a cui salta l'8080, 8085 o Z80.

L'indirizzo della locazione di memoria alla quale salta il microprocessore allorchè si introduce in comando Z si trova memorizzato nelle locazioni di memoria 000 313 e 000 314 (00CB e 00CC hex). *L'utente deve, perciò, memorizzare il byte basso del nuovo indirizzo a cui il microprocessore dovrà saltare nella locazione di memoria 000 313 (00CB hex) ed il byte alto del nuovo indirizzo alla locazione di memoria 000 314 (00CC hex).* Questo cambiamento può essere effettuato in un qualsiasi momento prima dell'esecuzione di TEA.

Avendo provveduto alla modifica dell'indirizzo, il microprocessore esegue un salto alla locazione di memoria specificata dall'utente al momento dell'introduzione del comando Z. Una volta eseguito il monitor di sistema o il debugger, uno di questi programmi può essere utilizzato per ripassare il controllo a TEA. Nel "saltare indietro" a TEA occorre procedere con attenzione, soprattutto nel caso che sia stata modificata la tabella di salto residente in memoria R/W. In quest'ultimo caso è necessario accertarsi che il salto avvenga nella locazione di memoria 000 030 (0018 hex) e non a quella 000 000 (0000 hex).

APPENDICE C

Aggiunta a TEA di un comando speciale dell'utente

Se lo si desidera, l'impiego del comando Z può servire a trasferire il controllo ad una nuova sezione di TEA o ad un'altro programma voluto dall'utente. Il comando Z non deve, in questo caso, essere utilizzato per trasferire il controllo del programma ad un monitor di sistema o ad un debugger. TEA comunque è stato scritto in modo che non soltanto sia possibile "personalizzare" il comando Z, ma sia anche possibile aggiungere un nuovo comando. Tale nuovo comando può essere definito sotto specie di *qualunque carattere singolo, che non sia uno di quelli già usati per i comandi di TEA.* I caratteri che *non possono* essere usati per il nuovo comando sono A, B, C, D, E, H, I, L, M, N, O, P, Q, R, S, X e Z. Può dunque essere usato qualunque carattere che non sia compreso tra questi. Una volta deciso quale sia il carattere che rappresenta il nuovo comando, il codice ASCII a 7 bit del carattere deve essere memorizzato nella locazione di memoria 021 361 (11F1 hex). Per aggiungere, ad esempio, il comando J, si deve memorizzare un 112 (4A hex) in questa locazioni di memoria.

Con l'introduzione del nuovo comando è altresì necessario memorizzare un'istruzione di salto in cui il secondo e terzo byte dell'istruzione stessa rappresentano la locazione di memoria alla quale si desidera trasferire il controllo. A questo effetto si deve memorizzare un 303 (C3 hex) alla locazione di memoria 000 320 (00D0 hex). Il byte basso dell'indirizzo di 16 bit deve quindi essere memorizzato in 000 321 (00D1 hex) e quello alto in 000 322 (00D2 hex). Una volta provveduto a ciò, il microprocessore salterà alla locazione di memoria specificata dall'istruzione di salto in tutte le occasioni in cui si introdurrà il comando aggiunto a TEA da parte dell'utente.

Se ne deduce che l'aggiunta di un comando particolare a TEA comporta un doppio provvedimento. Si deve, cioè, memorizzare in primo luogo il codice ASCII di 7 bit corrispondente al carattere che rappresenta il comando in memoria e si deve quindi memorizzare un'istruzione JMP di salto a tre byte. Spetta all'utente il determinare correttamente il codice ASCII da memorizzare così come l'individuare il secondo e terzo byte dell'istruzione JMP ed il caricamento in memoria dell'istruzione a tre byte.

APPENDICE D

Elenco riassuntivo dei comandi di TEA

Comandi di controllo

1. C=O Predisponi TEA nel modo di funzionamento ottale. Il listing generato da TEA nel corso della terza passata si varrà di numeri ottali ed i confini di memoria stampati in base al comando Q saranno in numeri ottali.

2. C=H Predisponi TEA nel modo di funzionamento esadecimale. Il listing generato da TEA nel corso della terza passata si varrà di numeri esadecimali ed i limiti di memoria stampati in base al comando Q saranno in numeri esadecimali.

3. C=M L'utilità di questo comando si evidenzia allorchè si vogliono cambiare i limiti di memoria del buffer del testo. All'introduzione di questo comando TEA pone la domanda "INITIAL ADDRESS?" e "FINAL ADDRESS?" ("indirizzo iniziale?" e "indirizzo finale?"). Si ricordi che, se non si desidera che uno o l'altro di questi indirizzi sia cambiato, è sufficiente premere il tasto RETURN dopo che la domanda è stata posta.

4. C=R La funzione di questo comando è quella di abilitare il dispositivo IOR (READER) (di lettura) in modo che sia possibile immettere nel text buffer un testo presente sul dispositivo di lettura, allorchè si introduce uno dei comandi A (APPEND), I (INSERT) oppure C (CHANGE).

5. C=B La funzione di questo comando è quella di permettere la lettura in memoria di un programma oggetto. Questo comando può essere introdotto in un qualunque momento per caricare un qualunque programma oggetto. In ogni caso, con la lettura in memoria del programma oggetto, si può avere l'alterazione del text buffer, di TEA o del suo stack.

6. C=Z Introducendo questo comando, il microprocessore ritorna ad un monitor di sistema ovvero ad un debugger. L'impiego di questo comando rende indispensabile una modifica di TEA. Si consulti, al proposito, l'Appendice B.

Comandi per l'Alterazione del Text Buffer

7. C=A Aggiunge al buffer del testo la parte di testo che si trova dopo questo comando. Quando, all'inizio, si dà il via a TEA, l'introduzione di questo comando permette di cominciare a creare il buffer. Se in quest'ultimo vi è già presente del testo, l'impiego di questo comando ha la funzione di consentire l'aggiunta di testo in coda a quello già contenuto dal buffer stesso.

8. C=I Inserisce una o più linee di testo a fronte della linea "n". È necessario, in questo comando, specificare un numero di linee valido. Quindi si preme il tasto RETURN.

C=15 ↓	Inserisci il seguente tasto a fronte della linea cinque.

9. C=C Sostituire la linea "n" con la (le) seguente (-i) linea (-e) di testo. Con questo comando occorre specificare un numero di linea valido. Si preme quindi il tasto RETURN.

C=C32 ↓	Cambia la linea 32 in modo che essa contenga il seguente testo (una o più linee di testo).

10. C=D La funzione di questo comando è quella di permettere di cancellare una o più linee di testo. Con questo comando è facoltativo specificare o meno uno o più numeri di linea validi. Quindi si preme di tasto RETURN.

C=D127 ↓	Cancella dal buffer la linea 127.
C=D34-59 ↓	Cancella dal buffer le linee da 34 a 59 compresa.
C=D ↓	Cancella l'intero buffer. Pur introducendo il comando, il buffer non risulta cancellato sino a che l'utente non risponde Y alla domanda "ARE YOU SURE?" (sei certo?).

Comandi di Accesso al Buffer del Testo

11. C=L La funzione di questo comando è quella di dar luogo al listing di una o più linee di testo. Con questo comando specificare uno o più numeri di linea validi è facoltativo. Una volta introdotto il (i) numero (-i) di linea si preme il tasto RETURN.

C=L34 ↓	Eseguire sul dispositivo IOO il listing relativo alla linea 34.
C=L341-349 ↓	Elenca sul dispositivo IOO le linee da 341 a 349 compresa.
C=L ↓	Realizza sul dispositivo IOO il listing dell'intero buffer.

12. C=P La funzione di questo comando serve ad ottenere sul dispositivo IOP la perforazione di una o più linee di testo. Per questo comando la citazione esplicita di uno o più numeri di linea è facoltativa. Una volta introdotto il (i) numero (-i) di linea si preme il tasto RETURN.

C=P19 ↓	Perfora la linea 19 sul dispositivo IOP.
C=P34-78 ↓	Perfora sul dispositivo IOP le linee da 34 a 78 comprese.
C=P ↓	Perfora sul dispositivo IOP l'intero buffer.

13. C=Q La funzione di questo comando è quella di dar luogo alla stampa dell'ultimo indirizzo utilizzato dal buffer, dell'ultima locazione di memoria a disposizione del buffer e del numero di linee di arresto contenute nel buffer. I comandi O e H stabiliranno il sistema di numerazione a cui il comando si dovrà attenere.

14. C=S La funzione di questo comando è quella di aiutare nella ricerca nel buffer di una stringa specificata dall'utente. Nel caso che si scopra una stringa uguale a quella cercata, segue la stampa dell'intera linea di testo.

C=S=STRING	(CTRL/S) Dopo che è stato introdotto S, TEA risponderà con un segno di uguale (=). Si introduce allora la "stringa campione", seguita da un CTRL/S.

15. C=N Continua la ricerca a partire dal numero di linea stampato in base al comando S, fino a scoprire ancora la stessa stringa oppure a giungere al termine del buffer. Se la stringa è scoperta di nuovo, si ottiene la stampa del relativo testo.

16. C=E Scambia la stringa specificata dall'utente con la stringa

rinvenuta in una linea di testo.

C=E=STRING	(CTRL/E) Scambia la stringa appena introdotta con la stringa rinvenuta nella linea di testo.
C=ES	Scambia la stringa precedentemente specificata mediante il comando E con la stringa rinvenuta nel buffer del testo.

17. C=X Assembla il programma sorgente contenuto (1) in memoria oppure (2) su banda. Per ulteriori particolari si veda il Capitolo 2.

APPENDICE E

Listing di TEA

In questa appendice è riportato il listing assemblato di TEA. Valendosi di questo listing, l'utente può caricare TEA in memoria R/W e programmare con esso delle EPROM. Il contenuto della locazione di memoria zero deve essere il primo valore da memorizzare (R/W o EPROM).

```
                /THIS IS THE TYCHON EDITOR-ASSEMBLER USED TO
                /CREATE ASSEMBLY LANGUAGE PROGRAMS FOR 8080-
                /AND 8085-BASED MICROCOMPUTERS.
                /COPYRIGHT 1979

                        DW TBLIM 031 177
                        DB LF 012
                        DB CR 015
                        DB TAB 011
                        DB STERM 000
                        DB BTERM 233
                        DB CTRLC 003

                        *030 000
18 00 C3        IOTEST, JMP     /GET THE INPUT DEVICE
18 01 51                INCHK   /DATA READY FLAG
18 02 06                0
18 03 C3        IONOEC, JMP     /INPUT A TTY OR CRT CHARACTER
18 04 6A                TTYI    /INTO A & B, BUT NO ECHO !
18 05 06                0
18 06 C3        IOECHO, JMP     /INPUT A CHARACTER INTO A & B
18 07 76                TTYIN   /BUT ECHO IT ON THE PRINTER.
18 08 06                0
18 09 C3        IOP,    JMP     /PUNCH OUT THE CHARACTER IN A
18 0A 79                TTYOUT  /THIS COULD BE A TTY PUNCH OR
18 0B 06                0       /AUDIO CASSETTE
18 0C C3        IOR,    JMP     /READ A CHARACTER INTO A&B
18 0D 56                IORDR   /FROM A TTY READER OR AUDIO CASSETTE
18 0E 06                0
18 0F C3        IOO,    JMP     /PRINT THE CHARACTER CONTAINED IN A.
18 10 79                TTYOUT  /THIS COULD BE A CRT, TTY OR CASSETTE.
18 11 06                0
18 12 00        BYTAB,  0       /THIS IS THE TABLE BETWEEN "SYMADF"
18 13 00                0       /AND "ENDRAM" FOR THE "DB" TABLE
18 14 00        HIADD,  0       /THIS IS THE HIGHEST ADDRESS THE
18 15 00                0       /THE TEXT BUFFER HAS USED.
18 16 00        ADCHK,  0       /0 IF NO ADDR. IS DEFINED, 377 IF ONE IS
18 17 00        INITLN, 0       /INITIAL LINE NUMBER SPECIFIED BY THE USER
18 18 00        SENSE,  0
18 19 00        FINLN,  0       /FINAL LINE NUMBER SPECIFIED BY THE USER
18 1A 00                0
18 1B 00        ADIFF,  0       /DIFFERENCE BETWEEN INITLN AND FINLN + 1
18 1C 00                0
18 1D 00        SRCADD, 0       /THIS IS USED BY N, NEXT OCCURENCE
18 1E 00                0
18 1F 00        STRFND, 0       /THIS IS USED BY THE SEARCH COMMAND
18 20 00        EXOK,   0       /THIS IS USED BY THE EXCHANGE COMMAND
18 21 00                0
18 22 00        TXTBUF, 0       /THE STARTING ADDRESS FOR THE
18 23 00                0       /TEXT BUFFER.
18 24 00        ENDRAM, 0       /LAST AVAILABLE R-W MEMORY LOCATION
18 25 00        ENDH,   0       /FOR THE TEXT BUFFER
18 26 00        TMP,    0
```

```
18 27 00          0
18 28 00  UNIT,   0        /THIS IS FOR THE DECIMAL
18 29 00          0        /LINE NUMBERS DURING THE
18 2A 00          0        /SEARCH ROUTINES.
18 2B 00  THOU,   0
18 2C 00  BLKU,   0        /THESE TWO MEMORY LOCATIONS ARE USED
18 2D 00  BLKT,   0        /FOR THE BLOCK NUMBER COUNTER.
18 2E 00  TFCLIN, 0        /TEXT, FIRST CHARACTER IN A LINE.
18 2F 00          0
18 30 00  TFMLIN, 0        /TEXT, FIRST MNEMONIC LETTER OF LINE
18 31 00          0
18 32 00  LINCNT, 0        /THE LINE COUNTER FOR ERROR MESSAGES
18 33 00          0
18 34 00  ADDCNT, 0        /THE "CURRENT ADDRESS COUNTER"
18 35 00          0
18 36 00  MNEVAL, 0        /MNEMONIC VALUE
18 37 00          0
18 38 00  PAGCNT, 0        /THIS IS THE # OF LINES PER PAGE
18 39 00  SYMADD, 0        /BEGINNING ADDRESS OF THE SYMBOL TABLE.
18 3A 00          0
18 3B 00  SYMADF, 0        /LAST LOC. USED BY THE SYMBOL TABLE.
18 3C 00          0
18 3D 00  MORT,   0        /FROM MEMORY OR TAPE ?
18 3E 00  HORO,   0        /HEX OR OCTAL LISTING BY TEA
18 3F 00  CHKSUM, 0        /CHECKSUM FOR PASS 2 OF THE ASSEMBLER
18 40 00  RORK,   0        /READER OR KEYBOARD INPUT TO THE EDITOR
18 41 00  PORL,   0        /SENSE REGISTER FOR PUNCH OR LIST
18 42 00  PASS,   0        /NUMBER OF THE PASS BEING PERFORMED
18 43 00  REG,    0        /SAVE THE H&L POINTERS TO THE REGISTER
18 44 00          0
18 45 00  TMPADD, 0
18 46 00          0
18 47 00  PAGNMB, 0        /CURRENT PAGE NUMBER FOR LISTINGS
18 48 00  NMBERR, 0        /NUMBER OF ERRORS ENCOUNTERED (1-256)
18 49 00  MTCHR,  0        /FOR ALPHABETIZING THE SYMBOL TABLE.
18 4A 00  BFUNIT, 0        /THESE 4 MEMORY LOCATIONS ARE USED TO
18 4B 00  BFTEN,  0        /STORE THE BCD LINE NUMBER THAT IS
18 4C 00  BFHUN,  0        /PRINTED OUT BY THE EDITOR DURING THE
18 4D 00  BFTHOU, 0        /APPEND, INSERT, CHANGE AND LIST CMDS.
18 4E 00  STRING, 0        /SEARCH STRING, UP TO 16 CHARACTERS.

                  *030 144
18 64 00  EXSTR,  0        /THE EXCHANGE STRING IS STORED HERE

                  *030 240
18 A0 00  STACK,  0        /THE STACK POINTER STARTS HERE.

                  *030 250
18 A8 00  BUFF2,  0        /THIS IS THE 1 LINE TEXT BUFFER
```

```
                    *000 000
00 00 00  START,    NOP      /THESE MEMORY LOCATIONS ARE FOR
00 01 00            NOP      /ANY USART INITIALIZATION
00 02 00            NOP      /INSTRUCTIONS
00 03 00            NOP
00 04 00            NOP
00 05 00            NOP
00 06 00            NOP
00 07 00            NOP
00 08 21            LXIH     /DUPLICATE THE I/O TABLE WITHIN
00 09 A5            IOTAB    /TEA IN R-W MEMORY STARTING FROM "IOTEST"
00 0A 07            0
00 0B 11            LXID     /D&E POINT TO THE DESTINATION
00 0C 00            IOTEST
00 0D 18            0
00 0E 06            MVIB     /NUMBER OF BYTES TO TRANSFER
00 0F 12            022      /022 = 18 DECIMAL (6 I/O ROUTINES)
00 10 7E  IOTRAN,   MOVAM    /GET THE DATA
00 11 12            STAXD    /STORE IT
00 12 23            INXH
00 13 13            INXD     /INCREMENT THE POINTERS
00 14 05            DCRB     /DONE ALL 18 MEMORY LOCATIONS ?
00 15 C2            JNZ      /NO, MOVE ANOTHER VALUE
00 16 10            IOTRAN
00 17 00            0
00 18 31  START1,   LXISP    /LOAD THE STACK POINTER WITH A
00 19 A0            STACK    /R/W MEMORY ADDRESS
00 1A 18            0
00 1B 2A  IADRIN,   LHLD     /GET WHATEVER ADDRESS IS STORED
00 1C 22            TXTBUF   /AS THE INITIAL ADDRESS OF THE
00 1D 18            0        /TEXT BUFFER
00 1E EB            XCHG     /GET IT INTO D&E.
00 1F 1B            DCXD     /DECREMENT IT BY ONE
00 20 21            LXIH     /LOAD H&L WITH THE ADDRESS OF
00 21 3C            IA       /THE MESSAGE "INITIAL ADDRESS ?"
00 22 12            0
00 23 CD            CALL     /PRINT "INITIAL ADDRESS", GET AN OCTAL OR
00 24 D4            ADCHK1   /HEX INITIAL ADDRESS AND CHECK
00 25 00            0        /ITS VALIDITY.
00 26 EB            XCHG     /ADDRESS IS VALID, GET IT INTO H&L
00 27 36            MVIM     /SAVE A CARRIAGE RETURN AT THIS ADDRESS
00 28 0D            CR
00 29 23            INXH     /INCREMENT THE ADDRESS
00 2A 22            SHLD     /SAVE THE ADDRESS AS THE BEGINNING
00 2B 22            TXTBUF   /OF THE TEXT BUFFER
00 2C 18            0
00 2D 2A  FADRIN,   LHLD     /LOAD H&L WITH ANY PREVIOUS FINAL
00 2E 24            ENDRAM   /ADDRESS
00 2F 18            0
00 30 EB            XCHG     /PUT IT INTO D&E
00 31 21            LXIH     /LOAD H&L WITH THE ADDRESS FOR THE
00 32 51            FA       /"FINAL ADDRESS"
00 33 12            0
00 34 CD            CALL     /PRINT THE MESSAGE, GET AN ADDRESS
```

```
00 35 D4          ADCHK1  /AND CHECK ITS VALIDITY
00 36 00          0
00 37 2A          LHLD    /NOW MAKE SURE THAT ITS GREATER
00 38 22          TXTBUF  /THAN THE INITIAL ADDRESS OF
00 39 18          0       /THE TEXT BUFFER
00 3A 7D          MOVAL
00 3B 93          SUBE
00 3C 7C          MOVAH
00 3D 9A          SBBD
00 3E D2          JNC     /D&E IS NOT GREATER THAN H&L
00 3F 2D          FADRIN  /SO GET ANOTHER FINAL ADDRESS
00 40 00          0
00 41 EB          XCHG
00 42 22          SHLD
00 43 24          ENDRAM  /SAVE THE LAST R-W LOCATION IN "ENDRAM"
00 44 18          0
00 45 AF          XRAA    /SET THE RORK SWITCH (READER OR
00 46 32          STA     /KEYBOARD) TO KEYBOARD ENTRY
00 47 40          RORK    /(BY MEANS OF THE IONOEC DEVICE)
00 48 18          0
00 49 21  YESNO,  LXIH
00 4A F7          SCIM    /PRINT THE MESSAGE
00 4B 11          0       /"SOURCE CONTAINED IN MEMORY ?"
00 4C CD          CALL
00 4D 7A          TOUT
00 4E 05          0
00 4F CD          CALL
00 50 06          IOECHO  /GET A CHARACTER WITH ECHO
00 51 18          0
00 52 FE          CPI
00 53 59          "Y"     /IF ANSWER IS YES, GO TO "NOTY"
00 54 C2          JNZ     /NOT A "Y", IS IT AN "N"?
00 55 60          NOTY
00 56 00          0
00 57 CD          CALL    /THERE IS A PROGRAM IN MEMORY,
00 58 26          CNTLN   /SO COUNT THE NUMBER OF LINES
00 59 01          0
00 5A 22          SHLD    /AND SAVE THE HIGHEST ADDRESS USED
00 5B 14          HIADD   /BY THE PROGRAM.
00 5C 18          0
00 5D C3          JMP     /THEN PRINT "C=".
00 5E 70          QUEST1
00 5F 00          0
00 60 FE  NOTY,   CPI     /NOT A "Y", IS IT AN "N"?
00 61 4E          "N"     /WAS A N FOR NO
00 62 C2          JNZ     /IT WAS NEITHER, TRY AGAIN
00 63 49          YESNO
00 64 00          0
00 65 CD  DELALL, CALL    /THE ANSWER WAS "N" FOR NO, SO RESET
00 66 2C          CLEAN   /A NUMBER OF POINTERS AND THEN
00 67 06          0
00 68 C3          JMP     /GET A USER COMMAND.
00 69 70          QUEST1
00 6A 00          0
```

```
00 6B 3E  QUEST,  MVIA    /TYPE A "?"
00 6C 3F          "?"
00 6D CD          CALL
00 6E 0F          IOO
00 6F 18          0
00 70 21  QUEST1, LXIH    /LOAD H&L WITH THE ADDRESS OF THE
00 71 64          CEQ     /MESSAGE "C="
00 72 12          0
00 73 CD          CALL    /PRINT THE "C="
00 74 7A          TOUT
00 75 05          0
00 76 3E  GETCHR, MVIA    /INITIALIZE THE MEMORY LOCATION "SENSE"
00 77 C0          300     /TO 300.  THIS IS USED BY THE "INPUT"
00 78 32          STA     /SUBROUTINE TO DETERMINE IF NO, ONE
00 79 18          SENSE   /OR TWO LINE NUMBERS WERE ENTERED.
00 7A 18          0
00 7B 3E          MVIA
00 7C 4C          "L"
00 7D 32          STA     /SET THE PUNCH/LIST DEVICE
00 7E 41          PORL    /TO LIST
00 7F 18          0
00 80 CD  COMDEC, CALL    /GET A USER COMMAND
00 81 06          IOECHO  /AND THEN "DECODE" IT.
00 82 18          0
00 83 21          LXIH    /SET H&L TO THE ASCII CHARACTER TABLE.
00 84 E0          CMDTAB
00 85 11          0
00 86 11          LXID    /SET D&E TO THE TABLE OF JUMP ADDRESSES
00 87 9D          CMDJMP
00 88 00          0
00 89 7E  NXTCMD, MOVAM   /GET AN ASCII TABLE CHARACTER
00 8A FE          CPI
00 8B 00          000     /END OF THE TABLE ?
00 8C 78          MOVAB   /IOECHO RETURNS THE CHARACTER IN A & B !
00 8D CA          JZ      /THE COMMAND IS NOT IN THE TABLE,
00 8E 6B          QUEST   /SO PRINT A ? AND THEN C=.
00 8F 00          0
00 90 BE          CMPM    /INPUT = TO THE TABLE ENTRY ?
00 91 CA          JZ      /YES, THEN GET THE ADDRESS TO JUMP TO
00 92 9B          CMDOK
00 93 00          0
00 94 23          INXH    /NO, SO POINT TO THE NEXT CHARACTER
00 95 13          INXD    /AND INCREMENT PAST THE JUMP IN-
00 96 13          INXD    /STRUCTION AND THE TWO ADDRESS
00 97 13          INXD    /BYTES
00 98 C3          JMP
00 99 89          NXTCMD
00 9A 00          0
00 9B EB  CMDOK,  XCHG    /GET THE ADDRESS INTO H&L
00 9C E9          PCHL    /AND GO TO THE PROPER ROUTINE

00 9D C3  CMDJMP, JMP
00 9E 20          DELET   /D=DELETE
00 9F 03          0
```

```
00 A0 C3          JMP
00 A1 B7          ASSEMB   /X=ASSEMB
00 A2 07          0
00 A3 C3          JMP
00 A4 75          LIST     /L=LIST
00 A5 02          0
00 A6 C3          JMP
00 A7 84          FINDIT   /S=SEARCH
00 A8 04          0
00 A9 C3          JMP
00 AA 70          TAPE     /P=PUNCH A TAPE
00 AB 02          0
00 AC C3          JMP
00 AD 85          MEMOUT   /Q=MEMORY LIMITS AND LAST LINE NUMBER
00 AE 05          0
00 AF C3          JMP
00 B0 20          CONSER   /N=CONTINUE SEARCH
00 B1 05          0
00 B2 C3          JMP
00 B3 FC          HEXOCT   /H OR O = SET THE LOCATION HORO
00 B4 00          0
00 B5 C3          JMP
00 B6 FC          HEXOCT   /SAME AS ABOVE
00 B7 00          0
00 B8 C3          JMP
00 B9 F5          READOK   /SET THE APPEND, INSERT AND CHANGE
00 BA 00          0        /INPUT TO THE READER DEVICE
00 BB C3          JMP
00 BC 03          APPEND   /A=APPEND
00 BD 01          0
00 BE C3          JMP
00 BF 6B          CHANGE   /C=CHANGE
00 C0 05          0
00 C1 C3          JMP
00 C2 6B          INSERT   /I=INSERT
00 C3 03          0
00 C4 C3          JMP
00 C5 9F          EXCHG    /E=EXCHANGE STRINGS
00 C6 06          0
00 C7 C3          JMP
00 C8 18          START1   /M=CHANGE MEMORY LIMITS
00 C9 00          0
00 CA C3          JMP      /Z=RETURN TO THE "SYSTEM MONITOR"
00 CB 18          START1   /OR "DEBUGGER"
00 CC 00          0
00 CD C3          JMP      /B=BOOTSTRAP; READ THE OBJECT PROGRAM
00 CE 52          BOOT     /PRODUCED BY THE SECOND PASS OF THE
00 CF 07          0        /ASSEMBLER INTO MEMORY
00 D0 00          0        /ONE USER COMMAND CAN FIT IN HERE
00 D1 00          0        /(MAKE SURE YOU ADD THE ASCII
00 D2 00          0        /VALUE FOR IT TO THE "CMDTAB" LIST)
00 D3 00          0

00 D4 E5  ADCHK1, PUSHH    /SAVE THE MESSAGE ADDRESS
```

```
00 D5 CD          CALL     /PRINT THE MESSAGE
00 D6 7A          TOUT
00 D7 05          0
00 D8 E1          POPH     /GET THE MESSAGE ADDRESS BACK
00 D9 CD          CALL     /GET AN OCTAL OR HEX 16-BIT
00 DA EB          ADRIN    /ADDRESS
00 DB 06          0
00 DC 01          LXIB     /LOAD B&C WITH THE "HI" ADDRESS
00 DD 7F          TBLIM
00 DE 19          0
00 DF CD          CALL
00 E0 F0          BCMDE    /THEN SUBTRACT D&E FROM B&C
00 E1 00          0
00 E2 D8          RC       /D&E > B&C, SO RETURN
00 E3 01          LXIB     /B&C > D&E, SO SEE IF D&E IS LESS
00 E4 00          START    /THAN THE STARTING ADDRESS OF
00 E5 00          0        /TEA
00 E6 CD          CALL     /SUBTRACT D&E FROM B&C
00 E7 F0          BCMDE
00 E8 00          0
00 E9 DA          JC       /THE ADDRESS IN D&E IS LESS THAN "TBLIM"
00 EA D4          ADCHK1   /BUT EQUAL TO OR GREATER THAN "START,"
00 EB 00          0        /SO GET ANOTHER ADDRESS.
00 EC CA          JZ
00 ED D4          ADCHK1
00 EE 00          0
00 EF C9          RET

00 F0 79  BCMDE,  MOVAC    /GET AN LSBY INTO THE A REGISTER
00 F1 93          SUBE     /SUBTRACT THE LSBY IN E
00 F2 78          MOVAB    /GET THE MSBY IN B
00 F3 9A          SBBD     /SUBTRACT-WITH-BORROW THE MSBY
00 F4 C9          RET      /IN D AND RETURN.

00 F5 78  READOK, MOVAB    /SET THE "RORK" REGISTER TO
00 F6 32          STA      /THE "READER" MODE FOR
00 F7 40          RORK     /THE A,C AND I COMMANDS
00 F8 18          0
00 F9 C3          JMP
00 FA 70          QUEST1
00 FB 00          0

00 FC 78  HEXOCT, MOVAB    /SAVE THE O OR H (ASCII CODE) IN THE
00 FD 32          STA
00 FE 3E          HORO     /MEMORY LOCATION "HORO"
00 FF 18          0
01 00 C3          JMP
01 01 70          QUEST1   /THEN GET ANOTHER USER COMMAND
01 02 00          0
```

```
                /THE APPEND COMMAND IS USED TO CREATE AN INITIAL
                /TEXT BUFFER.  WHEN WE START "TEA" THERE IS NO
                /TEXT BUFFER, SO WE APPEND NEW TEXT.

01 03 CD  APPEND, CALL    /TYPE A CR AND LF
01 04 39          CRLF
01 05 02          0
01 06 CD          CALL
01 07 26          CNTLN   /COUNT THE NUMBER OF LINES IN
01 08 01          0       /THE TEXT BUFFER
01 09 CD  AEND,   CALL    /GET ONE LINE OF TEXT IN THE
01 0A 3C          ONELN   /MINI-TEXT BUFFER
01 0B 01          0
01 0C 2A          LHLD    /NOW TRANSFER THE CONTENTS OF THE
01 0D 14          HIADD   /MINI-BUFFER TO THE MAIN TEXT BUFFER.
01 0E 18          0
01 0F 11          LXID
01 10 A8          BUFF2
01 11 18          0
01 12 1A  SAVBUF, LDAXD   /GET A CHARACTER FROM THE MINI-BUFFER
01 13 FE          CPI     /AT THE END OF THE MINI-BUFFER YET?
01 14 00          STERM
01 15 CA          JZ      /YES, THEN SAVE THE BUFFER TERMINATOR
01 16 1E          ENDBUF
01 17 01          0
01 18 77          MOVMA   /AND SAVE IT IN THE MAIN BUFFER
01 19 13          INXD
01 1A 23          INXH
01 1B C3          JMP
01 1C 12          SAVBUF
01 1D 01          0
01 1E 36  ENDBUF, MVIM
01 1F 9B          BTERM   /THEN SAVE A TEXT BUFFER TERMINATOR
01 20 22          SHLD
01 21 14          HIADD
01 22 18          0
01 23 C3          JMP
01 24 09          AEND    /NO, THEN GET ANOTHER LINE OF TEXT
01 25 01          0

01 26 CD  CNTLN,  CALL    /SET THE BUFFER LINE COUNT TO
01 27 68          BF1     /0001
01 28 04          0
01 29 2A          LHLD    /NOW COUNT THE NUMBER OF LINES
01 2A 22          TXTBUF  /IN THE TEXT BUFFER
01 2B 18          0
01 2C 7E  GETOND, MOVAM   /GET A CHARACTER
01 2D FE          CPI     /AT THE END OF THE BUFFER?
01 2E 9B          BTERM
01 2F C8          RZ      /YES, THEN RETURN WITH THE COUNT
01 30 23          INXH    /NO, INCREMENT THE MEMORY ADDRESS
01 31 FE          CPI     /CARRIAGE RETURN?
01 32 0D          CR
01 33 C2          JNZ     /NO, THEN CHECK THE NEXT CHARACTER
```

```
01 34 2C            GETOND
01 35 01            0
01 36 CD            CALL     /FOUND A CR, SO INCREMENT THE
01 37 5D            BFLINC   /THE BUFFER LINE INCREMENT SUB-
01 38 04            0        /ROUTINE
01 39 C3            JMP      /THEN GET ANOTHER COMMAND
01 3A 2C            GETOND   /FROM THE BUFFER AND TEST IT
01 3B 01            0

01 3C 2A  ONELN,    LHLD     /BEFORE A LINE CAN BE INPUT FROM THE
01 3D 14            HIADD    /IOO DEVICE, SEE IF THERE IS ANY MEMORY.
01 3E 18            0
01 3F EB            XCHG     /HIADD IN D&E
01 40 2A            LHLD     /GET THE ADDRESS OF THE LAST AVAILABLE
01 41 24            ENDRAM   /R/W MEMORY LOCATION
01 42 18            0
01 43 7D            MOVAL
01 44 93            SUBE
01 45 7C            MOVAH
01 46 9A            SBBD     /ARE WE TOO CLOSE?
01 47 CA            JZ       /YES, THEN PRINT THE "ME" ERROR
01 48 96            MMEXC    /MESSAGE
01 49 06            0
01 4A 21  ELINE,    LXIH     /LOAD REGISTER PAIR H WITH THE ADDRESS
01 4B A8            BUFF2    /WHERE THE ONE LINE OF TEXT WILL BE
01 4C 18            0        /STORED
01 4D 3A            LDA      /SHOULD WE GET THE CHARACTERS FROM
01 4E 40            RORK     /THE "IONOEC" DEVICE OR THE "IOR"
01 4F 18            0        /DEVICE?
01 50 FE            CPI
01 51 52            "R"
01 52 CA            JZ       /IT IS AN ASCII R, GET THE TEXT
01 53 43            RDRIN    /FROM THE "IOR" DEVICE
01 54 02            0
01 55 CD            CALL     /IF THERE IS SOME MEMORY, PRINT THE
01 56 73            BFOUT    /LINE NUMBER OF THE LINE TO BE
01 57 04            0        /ENTERED
01 58 0E  NEWTAB,   MVIC     /C= # OF SPACES PER TAB STOP
01 59 08            010
01 5A CD  NXTLET,   CALL
01 5B 03            IONOEC   /I-O DEVICE, INPUT TYPE, NO ECHO
01 5C 18            0
01 5D FE  NXTCIN,   CPI
01 5E 0D            CR       /CARRIAGE RETURN ENTERED?
01 5F CA            JZ
01 60 CB            CR       /YES, THEN THAT'S THE END OF THE LINE
01 61 01            0
01 62 FE            CPI      /WAS A CTRL/E ENTERED?
01 63 05            005
01 64 CA            JZ       /YES, THEN ERASE THE ENTIRE LINE
01 65 91            ALLBCK   /PRINT A CR & LF, PRINT THE LINE
01 66 01            0        /NUMBER AGAIN, AND GET SOME TEXT
01 67 FE            CPI      /LINE FEED ENTERED?
01 68 0A            LF
```

```
01 69 CA          JZ       /YES, THEN PRINT THE LINE THAT
01 6A 97          SHOW     /HAS BEEN ENTERED SO FAR.
01 6B 01          0
01 6C FE          CPI
01 6D 03          CTRLC    /CTRL-C?  THEN SEE WHATS UP.
01 6E CA          JZ
01 6F EF          NOGOOD
01 70 03          0
01 71 FE          CPI
01 72 09          TAB      /CTRL-TAB OR CTRL-I?
01 73 CA          JZ       /YES, THEN TYPE SPACES TO THE TAB STOP
01 74 25          TAB
01 75 02          0
01 76 FE          CPI
01 77 7F          177      /WAS IT THE RUBOUT KEY, FOR ERROR CORRECT.
01 78 CA          JZ       /YES, THEN ERASE A CHARACTER
01 79 D5          ERASE
01 7A 01          0
01 7B FE          CPI
01 7C 20          040      /IGNORE ANY ASCII CHARACTER WITH A
01 7D DA          JC       /CODE LESS THAN 040 EXCEPT AS ABOVE.
01 7E 5A          NXTLET
01 7F 01          0
01 80 FE          CPI
01 81 60          140      /NOTHING GREATER THAN 140 EITHER
01 82 D2          JNC
01 83 5A          NXTLET
01 84 01          0
01 85 77          MOVMA    /SAVE THE CHARACTER IN THE MINI-BUFFER
01 86 CD          CALL
01 87 0F          100      /OK, NOW TYPE THE CHARACTER OUT
01 88 18          0
01 89 23  NOEC,   INXH     /INCREMENT THE MINI-BUFFER POINTER
01 8A 0D          DCRC     /DECREMENT THE TAB STOP COUNTER
01 8B C2          JNZ      /GET ANOTHER CHARACTER IF ITS NOT 000
01 8C 5A          NXTLET
01 8D 01          0
01 8E C3          JMP      /SET THE SPACE (TAB) COUNTER BACK TO 010
01 8F 58          NEWTAB
01 90 01          0

01 91 CD  ALLBCK, CALL     /ERASE THE ENTIRE LINE, SO PRINT
01 92 39          CRLF     /A CR & LF, AND THEN REINITIALIZE
01 93 02          0        /THE ADDRESS IN REGISTER PAIR H
01 94 C3          JMP      /AND PRINT THE LINE NUMBER.
01 95 4A          ELINE
01 96 01          0

01 97 EB  SHOW,   XCHG     /PUT THE MINI-BUFFER ADDRESS IN D&E
01 98 21          LXIH     /GET THE MINI-BUFFER STARTING ADDRESS
01 99 A8          BUFF2·
01 9A 18          0
01 9B CD          CALL     /PRINT A CR & LF
01 9C 39          CRLF
```

```
01 9D 02            0
01 9E CD            CALL     /THEN PRINT THE CURRENT LINE NUMBER
01 9F 73            BFOUT
01 A0 04            0
01 A1 7B            MOVAE    /SUBTRACT THE STARTING ADDRESS OF
01 A2 95            SUBL     /MINI-BUFFER FROM THE "CURRENT
01 A3 5F            MOVEA    /BUFFER" ADDRESS TO DETERMINE THE
01 A4 7A            MOVAD    /NUMBER OF CHARACTERS TO PRINT.
01 A5 9C            SBBH
01 A6 57            MOVDA
01 A7 0E            MVIC     /SET THE SPACE COUNTER (FOR TABS)
01 A8 08            010
01 A9 C3            JMP      /THEN IS IF THE USER IS TRYING TO
01 AA C3            SHOWN    /"LIST" A LINE THAT CONTAINS
01 AB 01            0        /NO CHARACTERS
01 AC 7E  SHOWIT,   MOVAM    /GET A CHARACTER FROM THE BUFFER
01 AD FE            CPI      /IS IT THE CTRL/I CODE (TAB)?
01 AE 09            TAB
01 AF C2            JNZ      /NO, THEN SIMPLY PRINT
01 B0 B8            NOSHOW   /THE CHARACTER.
01 B1 01            0
01 B2 CD            CALL     /IT WAS A CTRL/I (TAB) SO PRINT
01 B3 2D            TABBR    /THE NUMBER OF SPACES DETERMINED
01 B4 02            0        /BY THE CONTENT OF THE C REGISTER
01 B5 C3            JMP      /THEN GET ANOTHER CHARACTER
01 B6 C1            SHOWT
01 B7 01            0
01 B8 CD  NOSHOW,   CALL     /NOT A CTRL/I, SO PRINT IT.
01 B9 0F            IOO
01 BA 18            0
01 BB 0D            DCRC     /DECREMENT THE SPACE COUNT
01 BC C2            JNZ      /NOT ZERO, SO SKIP OVER THE
01 BD C1            SHOWT    /MVIC INSTRUCTION
01 BE 01            0
01 BF 0E            MVIC     /C WAS DECREMENTED TO ZERO, SO
01 C0 08            010      /REINITIALIZE IT.
01 C1 23  SHOWT,    INXH     /INCREMENT THE BUFFER ADDRESS
01 C2 1B            DCXD     /DECREMENT THE CHARACTER COUNT
01 C3 7A  SHOWN,    MOVAD    /HAS THE COUNT BEEN DECREMENTED
01 C4 B3            ORAE     /TO ZERO?
01 C5 C2            JNZ      /NO, THEN GET ANOTHER CHARACTER
01 C6 AC            SHOWIT   /FROM THE MINI-BUFFER.
01 C7 01            0
01 C8 C3            JMP      /YES, THEN GET ANOTHER CHARACTER
01 C9 5A            NXTLET   /OR COMMAND FOR THE ONE LINE OF
01 CA 01            0        /TEXT.
```

```
01 CB 77  CR,     MOVMA   /RETURN KEY PRESSED, SAVE IT
01 CC 23          INXH    /INCREMENT THE MINI-BUFFER ADDRESS
01 CD 36          MVIM    /SAVE A BUFFER TERMINATOR AFTER
01 CE 00          STERM   /THE CR (015, OD) VALUE.
01 CF CD          CALL    /INCREMENT THE BUFFER LINE
01 D0 5D          BFLINC  /NUMBER
01 D1 04          0
01 D2 C3          JMP     /PRINT A CR AND LF AND THEN
01 D3 39          CRLF    /RETURN FROM THE "ONELN"
01 D4 02          0       /SUBROUTINE

01 D5 3E  ERASE,  MVIA    /TYPE A BACKWARDS SLASH
01 D6 DC          334     /ON THE IOO DEVICE.
01 D7 CD          CALL
01 D8 0F          IOO
01 D9 18          0
01 DA 2B  E1,     DCXH    /DECREMENT THE MINI-BUFFER POINTER
01 DB EB          XCHG
01 DC 21          LXIH    /THIS PREVENTS US FROM "BACKING UP"
01 DD A8          BUFF2   /INTO ANOTHER LINE OF TEXT.
01 DE 18          0
01 DF 7B          MOVAE
01 E0 95          SUBL
01 E1 7A          MOVAD   /CALCULATE THE NUMBER OF CHARACTERS
01 E2 9C          SBBH    /IN THE MINI TEXT BUFFER
01 E3 EB          XCHG
01 E4 D2          JNC     /OK, WE HAVEN'T BACKED UP ALL THE WAY
01 E5 EB          NOBEG   /TO THE BEGINNING OF THE BUFFER,
01 E6 01          0       /FIND OUT WHAT CHARACTER WAS JUST ERASED
01 E7 23          INXH
01 E8 C3          JMP     /THIS IS TO PREVENT ECHOING
01 E9 1A          NOOUT   /OF THE FIRST CHARACTER IN
01 EA 02          0       /A LINE UPON MULTIPLE ERASURES
01 EB 7E  NOBEG,  MOVAM   /GET A CHARACTER, WAS IT A "TAB" ?
01 EC FE          CPI
01 ED 09          TAB
01 EE C2          JNZ     /NO, THEN ITS A CHARACTER
01 EF 0D          NOCT
01 F0 02          0
01 F1 E5          PUSHH   /IT WAS A TAB, SAVE THE BUFFER ADDRESS
01 F2 0E  BACK1,  MVIC    /SET THE "SPACE COUNTER" TO 010
01 F3 08          010
01 F4 2B  BACK2,  DCXH    /DECREMENT THE BUFFER ADDRESS
01 F5 7E          MOVAM   /AND SEE IF WE HAVE DECREMENTED
01 F6 FE          CPI     /TO THE VERY BEGINNING OF THE LINE
01 F7 0D          CR
01 F8 CA          JZ      /THE ENTIRE LINE HAS BEEN ERASED
01 F9 07          TABX    /(WE HAVE BACKED UP TO THE BEGINNING
01 FA 02          0       /OF THE LINE), SO GET A CHARACTER
01 FB FE          CPI     /FOUND ANOTHER CONSECUTIVE TAB?
01 FC 09          TAB
01 FD CA          JZ      /YES, THEN GET ANOTHER CHARACTER
01 FE 07          TABX
01 FF 02          0
```

```
02 08 0D          DCRC    /DECREMENT THE COUNT
02 01 C2          JNZ     /IF ITS NOT ZERO, GET ANOTHER
02 02 F4          BACK2   /CHARACTER.
02 03 01          0
02 04 C3          JMP     /IT IS ZERO, REINITIALIZE C, AND
02 05 F2          BACK1   /THEN GET ANOTHER CHARACTER.
02 06 01          0
02 07 E1  TABX,   POPH    /GET THE TEXT BUFFER ADDRESS INTO H&L
02 08 3E          MVIA    /AND TYPE OUT
02 09 A0          240     /A SPACE EACH TIME A TAB IS
02 0A C3          JMP     /FOUND.
02 0B 17          NOET1   /AND GET A CHARACTER FROM "IOO"
02 0C 02          0

02 0D 0C  NOCT,   INRC    /NOT A TAB, INCREMENT THE SPACE COUNTER
02 0E 79          MOVAC   /GET THE COUNTER
02 0F FE          CPI     /HAS IT BEEN INCREMENTED PAST ITS
02 10 09          011     /UPPER LIMIT?
02 11 C2          JNZ     /NO, THEN GET ANOTHER CHARACTER.
02 12 16          NOET
02 13 02          0
02 14 0E          MVIC    /YES, SO SET C TO ZERO
02 15 00          000
02 16 7E  NOET,   MOVAM   /GET THE "ERASED" CHARACTER
02 17 CD  NOET1,  CALL    /PRINT IT ON THE I/O OUTPUT DEVICE.
02 18 0F          IOO
02 19 18          0
02 1A CD  NOOUT,  CALL    /GET A CHARACTER FROM THE KEYBOARD
02 1B 03          IONOEC  /A NEW BUFFER CHARACTER OR
02 1C 18          0       /ERASE COMMAND.
02 1D FE          CPI     /ANOTHER ERASE COMMAND ?
02 1E 7F          177
02 1F CA          JZ      /YES, THEN PRINT THE "ERASED"
02 20 DA          E1      /CHARACTER.
02 21 01          0
02 22 C3          JMP     /NOT AN ERASE COMMAND, SO ADD THE
02 23 5D          NXTCIN  /CHARACTER TO THE BUFFER OR
02 24 01          0       /INTERPRETE IT AS A COMMAND.

02 25 77  TAB,    MOVMA   /SAVE THE TAB IN MEMORY
02 26 23          INXH    /INCREMENT THE BUFFER ADDRESS
02 27 CD          CALL    /PRINT THE NUMBER OF SPACES,
02 28 2D          TABBR   /AS DETERMINED BY THE CONTENT
02 29 02          0       /OF THE C REGISTER
02 2A C3          JMP     /THEN GET ANOTHER CHARACTER FOR
02 2B 58          NEWTAB  /THE TEXT BUFFER OR ANOTHER
02 2C 01          0       /COMMAND

02 2D 3E  TABBR,  MVIA    /LOAD THE A REGISTER WITH THE
02 2E A0          240     /ASCII VALUE FOR A "SPACE"
02 2F CD          CALL    /PRINT THE SPACE ON THE I/O
02 30 0F          IOO     /OUTPUT DEVICE
02 31 18          0
02 32 0D          DCRC    /DECREMENT THE SPACE COUNT IN C
```

```
02 33 C2          JNZ     /IF THE RESULT IS NON-ZERO,
02 34 2D          TABBR   /PRINT ANOTHER SPACE.
02 35 02          0
02 36 0E          MVIC    /PRINTED THE SPACES, REINITILIZE THE
02 37 08          010     /C REGISTER
02 38 C9          RET     /AND THEN RETURN

02 39 3E  CRLF,   MVIA    /LOAD THE A REGISTER WITH THE ASCII
02 3A 8D          215     /VALUE FOR A CARRIAGE RETURN
02 3B CD          CALL    /PRINT THE CHARACTER ON THE I/O
02 3C 0F          IOO     /OUTPUT DEVICE.
02 3D 18          0
02 3E 3E          MVIA    /THEN LOAD THE A REGISTER WITH THE
02 3F 8A          212     /ASCII VALUE FOR A LINE FEED
02 40 C3          JMP     /AND PRINT IT ON THE I/O OUTPUT
02 41 0F          IOO     /DEVICE
02 42 18          0

          /"RDRIN" IS USED TO READ A LINE OF TEXT INTO MEMORY
          /ADDRESSED BY REGISTER PAIR H.  IF A CR IS READ, THE
          /MICROCOMPUTER RETURNS TO "ONELN." IF A CTRL/C IS
          /READ, CONTROL RETURNS TO THE COMMAND DECODER.

02 43 CD  RDRIN,  CALL    /GET A CHARACTER FROM THE READER
02 44 0C          IOR
02 45 18          0
02 46 FE          CPI     /WAS A CTRL/I (TAB) JUST READ?
02 47 09          TAB
02 48 CA          JZ      /YES, THEN SAVE IT IN MEMORY
02 49 5F          SAVRC   /AND READ ANOTHER CHARACTER.
02 4A 02          0
02 4B FE          CPI     /WAS A CARRIAGE RETURN JUST READ?
02 4C 0D          CR
02 4D CA          JZ      /YES, THEN SAVE IT IN MEMORY AND
02 4E 5F          SAVRC   /READ ANOTHER CHARACTER.
02 4F 02          0
02 50 FE          CPI     /WAS A CTRL/C JUST READ?
02 51 03          CTRLC
02 52 CA          JZ      /YES, THEN WE ARE DONE READING
02 53 69          DONRDR  /FROM THE READER.
02 54 02          0
02 55 FE          CPI     /WAS THE CHARACTER READ LESS THAN 040?
02 56 20          040
02 57 DA          JC      /YES, THEN IGNORE THE CHARACTER
02 58 43          RDRIN
02 59 02          0
02 5A FE          CPI     /WAS THE CHARACTER GREATER THAN 137?
02 5B 60          140
02 5C D2          JNC     /YES, THEN IGNORE THE CHARACTER
02 5D 43          RDRIN
02 5E 02          0
02 5F 77  SAVRC,  MOVMA   /THE CHARACTER IS "OK", SO SAVE IT
02 60 23          INXH    /INCREMENT THE MEMORY ADDRESS.
02 61 FE          CPI     /WAS THE CHARACTER A CARRIAGE RETURN?
```

```
02 62 0D            CR
02 63 C2            JNZ      /NO, THEN READ ANOTHER CHARACTER.
02 64 43            RDRIN
02 65 02            0
02 66 36            MVIM     /YES, THEN SAVE A ZERO AFTER THE
02 67 00            000      /ASCII VALUE FOR A CARRIAGE RETURN
02 68 C9            RET      /RETURN FROM "ONELN"

02 69 AF  DONRDR,   XRAA     /FOUND THE CTRL/C AT THE END OF
02 6A 32            STA      /THE TAPE, SO DISABLE THE READER
02 6B 40            RORK     /(ENABLE THE KEYBOARD) AND RETURN
02 6C 16            0
02 6D C3            JMP      /TO THE COMMAND DECODER
02 6E EF            NOGOOD
02 6F 03            0

                    /THIS ROUTINE PUNCHES A PAPER TAPE
                    /WITH A CTRL-C AT THE END.

02 70 3E  TAPE,     MVIA     /SET THE "PORL" SWITCH TO P FOR
02 71 50            "P"      /PUNCHING A PAPER TAPE
02 72 C3            JMP      /THEN SAVE THE CONTENT OF THE
02 73 7A            SETPL    /A REGISTER IN "PORL"
02 74 02            0
02 75 CD  LIST,     CALL     /SET THE BUFFER LINE NUMBER
02 76 68            BFI      /TO 0001
02 77 04            0
02 78 3E            MVIA     /SET THE PORL SENSE REGISTER TO
02 79 4C            "L"      /L FOR A LIST OF THE TEXT
02 7A 32  SETPL,    STA      /SAVE THE ASCII P OR L IN MEMORY
02 7B 41            PORL
02 7C 16            0
02 7D CD  LORP,     CALL     /FIND OUT IF WE ARE TO LIST OR PUNCH
02 7E C2            INPUT    /ONE LINE, A GROUP OF LINES OR THE
02 7F 03            0        /CONTENT OF THE ENTIRE TEXT BUFFER.
02 80 FE            CPI      /PUNCH OR LIST ONE LINE?
02 81 00            000
02 82 CA            JZ       /YES
02 83 A5            ONELST   /IF A =000, DO 1 LINE
02 84 02            0
02 85 FE            CPI      /PUNCH OR LIST THE ENTIRE TEXT BUFFER?
02 86 02            002
02 87 CA            JZ       /YES
02 88 AE            ALLIST   /IF A=002, DO COMPLETE BUFFER
02 89 02            0
02 8A CD            CALL     /NO, WE'RE GOING TO PUNCH OR LIST
02 8B 90            DIFF     /A GROUP OF LINES, SO CALCULATE
02 8C 02            0        /THE NUMBER OF LINES.
02 8D C3            JMP      /THEN PUNCH OR LIST THE GROUP
02 8E A8            BLKLST   /OF LINES IN THE TEXT BUFFER
02 8F 02            0

02 90 2A  DIFF,     LHLD     /CALCULATE THE DIFFERENCE BETWEEN
02 91 17            INITLN   /"INITLN" AND "FINLN"
```

```
02 92 18            0
02 93 EB            XCHG
02 94 2A            LHLD     /THESE CONTAIN THE BINARY EQUIVALENTS
02 95 19            FINLN    /OF THE NUMBERS THE USER TYPED IN
02 96 18            0
02 97 7D  TAKEDF,   MOVAL    /SUBTRACT THE 16-BIT "INITLN" FROM
02 98 93            SUBE     /THE 16-BIT "FINLN"
02 99 6F            MOVLA
02 9A 7C            MOVAH
02 9B 9A            SBBD
02 9C 67            MOVHA
02 9D 23            INXH     /INCREMENT THE DIFFERENCE BY 1
02 9E 22            SHLD     /IF WE TYPED L3-5, WE WANT TO LIST
02 9F 1B            ADIFF    /LINES 3,4 AND 5 OR LIST 3 LINES !
02 A0 18            0
02 A1 D0            RNC      /RETURN IF "INITLN" IS LESS THAN
02 A2 C3            JMP      /"FINLN". OTHERWISE, PRINT A
02 A3 EF            NOGOOD   /? AND RETURN TO THE COMMAND DECODER.
02 A4 03            0

02 A5 CD  ONELST,   CALL     /LIST OR PUNCH ONE LINE, SO
02 A6 63            DIFF1    /SET ADIFF TO 000 001
02 A7 03            0
02 A8 CD  BLKLST,   CALL     /NOW FIND THE BEGINNING OF THE LINE
02 A9 32            SEEK     /IN THE TEXT BUFFER USING "INITLN" AS THE
02 AA 04            0        /CARRAGE RETURN OR LINE COUNTER
02 AB C3            JMP
02 AC B8            LDROK    /SHOULD WE PUNCH LEADER ?
02 AD 02            0

02 AE 2E  ALLIST,   MVIL     /WE WANT TO LIST OR PUNCH THE CONTENT
02 AF FF            377      /OF THE ENTIRE TEXT BUFFER.
02 B0 26            MVIH
02 B1 FF            377
02 B2 22            SHLD     /SO SET "ADIFF" OR DIFFERENCE TO
02 B3 1B            ADIFF    /377 377
02 B4 18            0
02 B5 2A            LHLD     /LOAD REGISTER PAIR H WITH THE
02 B6 22            TXTBUF   /STARTING ADDRESS OF THE TEXT BUFFER
02 B7 18            0
02 B8 3A  LDROK,    LDA      /IS THIS A PUNCH OR LIST OPERATION?
02 B9 41            PORL
02 BA 18            0
02 BB FE            CPI
02 BC 50            "P"
02 BD C2            JNZ      /ITS A LIST OPERATION
02 BE C6            FRSTLN
02 BF 02            0
02 C0 CD            CALL     /ITS A PUNCH OPERATION, SO
02 C1 55            LDR      /PUNCH SOME LEADER
02 C2 05            0
02 C3 C3            JMP      /AND THEN START PUNCHING THE TAPE
02 C4 3D            TAPE1
02 C5 05            0
```

```
02 C6 3A  FRSTLN, LDA
02 C7 41          PORL     /DO WE PUNCH THE BUFFER OR LIST IT ?
02 C8 18          0
02 C9 FE          CPI
02 CA 50          "P"
02 CB CA          JZ       /P FOR PUNCH, SO PUNCH IT.
02 CC 3D          TAPE1
02 CD 05          0
02 CE CD          CALL     /WE LIST IT, SO PRINT A LINE NUMBER
02 CF 73          BFOUT    /FOR THE LINE(S) OF TEXT.
02 D0 04          0
02 D1 CD          CALL     /THEN INCREMENT THE LINE NUMBER BY 1
02 D2 5D          BFLINC
02 D3 04          0
02 D4 0E  SCHAR,  MVIC     /LOAD THE C REGISTER WITH 010 (THE
02 D5 08          010      /SPACE COUNT) IN CASE A "TAB" IS FOUND.
02 D6 7E  LCHAR,  MOVAM    /GET A CHARACTER FROM THE TEXT BUFFER
02 D7 FE          CPI
02 D8 9B          BTERM    /BUFFER TERMINATOR YET?
02 D9 CA          JZ       /YES, THEN GET ANOTHER COMMAND.
02 DA 70          QUEST1
02 DB 00          0
02 DC FE          CPI
02 DD 0D          CR       /CARRIAGE RETURN ?
02 DE CA          JZ       /YES, THEN DECREMENT THE LINE COUNT.
02 DF F9          CNTCRS
02 E0 02          0
02 E1 FE          CPI      /WAS A CTRL/I (TAB) READ?
02 E2 09          TAB
02 E3 CA          JZ       /YES, THEN PRINT THE REQUIRED
02 E4 F3          TAB1     /NUMBER OF SPACES.
02 E5 02          0
02 E6 CD          CALL     /NONE OF THE ABOVE, SO PRINT IT.
02 E7 0F          IOO
02 E8 18          0
02 E9 0D          DCRC     /DECREMENT THE "SPACE" COUNTER
02 EA C2          JNZ      /IF ITS NOT ZERO, DO NOT
02 EB EF          RET2     /REINITIALIZE IT.
02 EC 02          0
02 ED 0E          MVIC     /IT IS ZERO, SO REINITIALIZE IT
02 EE 08          010
02 EF 23  RET2,   INXH     /INCREMENT THE MEMORY ADDRESS
02 F0 C3          JMP      /AND GET ANOTHER CHARACTER
02 F1 D6          LCHAR
02 F2 02          0

02 F3 CD  TAB1,   CALL     /A TAB WAS FOUND IN MEMORY DURING THE
02 F4 2D          TABBR    /LISTING OPERATION, SO PRINT THE
02 F5 02          0        /REQUIRED NUMBER OF SPACES
02 F6 C3          JMP      /AND GET ANOTHER CHARACTER
02 F7 EF          RET2
02 F8 02          0

02 F9 CD  CNTCRS, CALL     /FOUND A CR, SO PRINT A CR & LF
```

```
02 FA 39          CRLF
02 FB 02          0
02 FC 23  ENDPL,  INXH     /INCREMENT THE BUFFER ADDRESS
02 FD E5          PUSHH    /SAVE THE BUFFER ADDRESS ON THE STACK
02 FE 2A          LHLD     /LOAD H&L WITH THE NUMBER OF LINES
02 FF 1B          ADIFF    /TO BE LISTED OR PUNCHED.
03 00 18          0
03 01 2B          DCXH     /DECREMENT THIS NUMBER
03 02 22          SHLD     /AND SAVE IT BACK IN R/W MEMORY
03 03 1B          ADIFF
03 04 18          0
03 05 7C          MOVAH    /IS THE RESULT OF THE DCXH ZERO?
03 06 B5          ORAL
03 07 E1          POPH     /POP THE BUFFER ADDRESS OFF OF THE STACK
03 08 C2          JNZ      /THE RESULT IS NON-ZERO, SO THERE ARE
03 09 C6          FRSTLN   /ADDITIONAL LINES TO LIST OR PUNCH
03 0A 02          0
03 0B 21          LXIH     /DONE LISTING OR PUNCHING, LOAD H&L
03 0C 41          PORL     /WITH THE ADDRESS OF "PORL"
03 0D 18          0
03 0E 3E          MVIA     /LOAD A WITH THE ASCII VALUE FOR THE
03 0F 4C          "L"      /L OR LIST OPERATION
03 10 BE          CMPM     /COMPARE IT TO THE COMMAND
03 11 77          MOVMA    /THEN SET "PORL" BACK TO THE L MODE
03 12 CA          JZ       /WE WERE LISTING, SO JUMP BACK TO
03 13 70          QUEST1   /TO THE COMMAND DECODER
03 14 00          0
03 15 3E  TRLR,   MVIA     /WE WERE PUNCHING, SO PUNCH
03 16 83          203      /A CTRL/C AT THE END OF THE TAPE
03 17 CD          CALL
03 18 09          IOP
03 19 18          0
03 1A CD          CALL     /FOLLOWED BY TRAILER.
03 1B 55          LDR
03 1C 05          0
03 1D C3          JMP      /THEN GET ANOTHER COMMAND.
03 1E 70          QUEST1
03 1F 00          0
```

```
          /THIS PORTION OF THE PROGRAM IS EXECUTED IF WE WANT TO
          /DELETE ONE LINE, A GROUP OF LINES OR THE CONTENT OF THE
          /ENTIRE TEXT BUFFER.

03 20 CD  DELET,  CALL    /GET THE LINE NUMBER(S) TO BE DELETED.
03 21 C2          INPUT
03 22 03          0
03 23 FE          CPI     /DELETING ONE LINE?
03 24 00          000
03 25 CA          JZ
03 26 3E          DEONE   /YES, THEN JUMP TO DEONE
03 27 03          0
03 28 FE          CPI     /DELETING THE ENTIRE TEXT BUFFER?
03 29 02          002
03 2A C2          JNZ     /NO, THEN WE ARE DELETING A
03 2B 44          BLKD    /GROUP (BLOCK) OF LINES.
03 2C 03          0
03 2D 21          LXIH    /TRYING TO DELETE EVERYTHING, SO
03 2E 17          AYS     /TYPE OUT "ARE YOU SURE ?"
03 2F 12          0       /(LOAD H&L WITH THE MESSAGE ADDRESS)
03 30 CD          CALL    /PRINT THE MESSAGE
03 31 7A          TOUT
03 32 05          0
03 33 CD          CALL    /GET THE USER'S RESPONSE
03 34 06          IOECHO
03 35 18          0
03 36 FE          CPI     /IS THE ANSWER Y FOR YES?
03 37 59          "Y"     /(Y=YES=DELETE THE ENTIRE BUFFER)
03 38 C2          JNZ     /NOT A "Y", LEAVE THE TEXT BUFFER ALONE
03 39 6B          QUEST   /AND GET ANOTHER USER COMMAND
03 3A 00          0
03 3B C3          JMP     /A "Y" WAS ENTERED, SO DELETE THE
03 3C 65          DELALL  /ENTIRE TEXT BUFFER.
03 3D 00          0

03 3E CD  DEONE,  CALL    /DELETING ONE LINE, SET "ADIFF"
03 3F 63          DIFF1   /TO 000 001.
03 40 03          0
03 41 C3          JMP     /THEN DELETE THE LINE FROM THE
03 42 47          BLKD1   /TEXT BUFFER.
03 43 03          0

03 44 CD  BLKD,   CALL    /DELETING A BLOCK (GROUP) OF LINES, SO
03 45 90          DIFF    /SUBTRACT "INITLN" FROM "FINLN" AND
03 46 02          0       /ADD ONE TO THE RESULT.
03 47 CD  BLKD1,  CALL    /DELETE THE GROUP OF LINES
03 48 4D          DELIT
03 49 03          0
03 4A C3          JMP     /THEN GO BACK TO THE COMMAND DECODER.
03 4B 70          QUEST1
03 4C 00          0

03 4D CD  DELIT,  CALL    /H&L WILL POINT TO THE FIRST CHAR-
03 4E 32          SEEK    /ACTER THE NUMBER OF CR'S SPECIFIED
```

```
03 4F 04            0        /BY "INITLN" WHEN CONTROL RETURNS
03 50 CD            CALL
03 51 4F            SEEK1    /USE D&E STARTING AT H&L
03 52 04            0        /UNTIL THE DIFFERENCE IS FOUND.
03 53 1A  POPMOR,   LDAXD    /THEN MOVE THE TEXT DOWN.
03 54 77            MOVMA    /MOVE ONE CHARACTER DOWN
03 55 FE            CPI
03 56 9B            BTERM    /BUFFER TERMINATOR?
03 57 CA            JZ       /YES, WE ARE DONE
03 58 5F            ENDPOP
03 59 03            0
03 5A 13            INXD     /NO, INCREMENT THE TWO ADDRESSES
03 5B 23            INXH
03 5C C3            JMP      /AND MOVE ANOTHER CHARACTER.
03 5D 53            POPMOR
03 5E 03            0

03 5F 22  ENDPOP,   SHLD     /MOVED ALL OF THE CHARACTERS, SAVE
03 60 14            HIADD    /THE HIGHEST ADDRESS USED.
03 61 18            0
03 62 C9            RET      /THEN RETURN FROM "DELIT"

03 63 2E  DIFF1,    MVIL     /SET THE L REGISTER TO 001
03 64 01            001
03 65 26            MVIH     /SET THE H REGISTER TO 000
03 66 00            000
03 67 22            SHLD     /SAVE 000 001 IN "ADIFF"
03 68 1B            ADIFF
03 69 18            0
03 6A C9            RET      /AND THEN RETURN

          /TO INSERT, WE GET A LINE NUMBER TO INSERT IN FRONT OF
          /THEN WE FIND THE H&L ADDRESS FOR THAT LINE NUMBER.
          /WE THEN INPUT 1 LINE OF CODE INTO THE MINI-BUFFER
          /AND THEN COUNT HOW MANY CHARACTERS ARE IN THE LINE.
          /THE MAIN TEXT BUFFER IS THEN "OPENED UP", MAKING
          /ROOM FOR THE ONE LINE OF TEXT.  FINALLY, THE
          /ONE LINE OF TEXT IS MOVED FROM THE MINI-BUFFER TO
          /THE MAIN BUFFER AND THEN WE SEE IF ANOTHER LINE
          /WILL BE TYPED IN BY THE USER.

03 6B CD  INSERT,   CALL     /GET THE LINE NUMBER TO INSERT
03 6C C2            INPUT    /NEW TEXT IN FRONT OF.
03 6D 03            0
03 6E CD            CALL     /WAS 1 (CR) OR 1 3-5 ENTERED?
03 6F 26            CHKIT    /IF SO, ERROR, BECAUSE ONLY ONE
03 70 06            0        /LINE NUMBER CAN BE SPECIFIED (1 4).
03 71 CD  CHNG1,    CALL     /SET THE LINE BUFFER NUMBER
03 72 68            BF1      /TO 0001
03 73 04            0
03 74 CD            CALL     /THEN FIND THE POINT OF INSERTION
03 75 32            SEEK     /IN THE MAIN TEXT BUFFER.  THE
03 76 04            0        /LINE # IS INCRED WITH EACH CR.
03 77 22  NXTIN,    SHLD     /SAVE THE INSERTION POINT ADDRESS
```

```
03 78 17          INITLN  /IF ADDITIONAL LINES MUST BE
03 79 18          0       /INSERTED.
03 7A CD          CALL
03 7B 3C          ONELN   /INPUT 1 LINE INTO THE MINI-BUFFER
03 7C 01          0
03 7D 21          LXIH    /LOAD H&L WITH THE ADDRESS WHERE
03 7E A8          BUFF2   /THE ONE LINE OF TEXT IS STORED.
03 7F 18          0
03 80 CD          CALL    /NOW STORE THE LINE OF TEXT
03 81 86          STOBUF  /IN THE MAIN TEXT BUFFER
03 82 03          0
03 83 C3          JMP     /THEN GET ANOTHER LINE OF TEXT
03 84 77          NXTIN   /TO BE INSERTED
03 85 03          0

03 86 E5  STOBUF, PUSHH   /SAVE THE BUFFER ADDRESS
03 87 1E          MVIE    /LOAD D&E WITH 000 000 SO THAT THE
03 88 00          000     /NUMBER OF CHARACTERS TO BE INSERTED
03 89 16          MVID    /CAN BE COUNTED.
03 8A 00          000
03 8B 7E  CNTCHR, MOVAM   /GET A CHAR. FROM THE LINE OF TEXT
03 8C FE          CPI     /FOUND THE END OF THE LINE OF
03 8D 00          STERM   /TEXT YET?
03 8E CA          JZ      /YES, THEN D&E CONTAIN THE
03 8F 96          ENDCNT  /NUMBER OF CHARACTERS IN THE LINE
03 90 03          0
03 91 23          INXH    /NOT THE LINE TERMINATOR, INCREMENT
03 92 13          INXD    /THE ADDRESS AND THE COUNT.
03 93 C3          JMP     /THEN GET THE NEXT CHARACTER AND
03 94 8B          CNTCHR  /COMPARE IT TO "STERM"
03 95 03          0
03 96 2A  ENDCNT, LHLD    /GET THE ADDRESS FOR THE POINT
03 97 17          INITLN  /OF INSERTION.
03 98 18          0
03 99 E5          PUSHH
03 9A C1          POPB    /MOVE THIS ADDRESS TO B&C
03 9B 2A          LHLD
03 9C 14          HIADD   /GET ADDRESS OF THE 233 AT THE END
03 9D 18          0       /OF THE MAIN TEXT BUFFER
03 9E E5          PUSHH   /SAVE THIS ADDRESS
03 9F 7D          MOVAL   /SUBTRACT THE "POINT OF INSERTION"
03 A0 91          SUBC    /ADDRESS FROM THE HIGHEST ADDRESS
03 A1 4F          MOVCA   /USED.
03 A2 7C          MOVAH
03 A3 98          SBBB
03 A4 47          MOVBA
03 A5 03          INXB    /INCREMENT THE NUMBER OF BYTES BY 1
03 A6 19          DADD    /HIADD + CHARACTER COUNT IN H&L
03 A7 22          SHLD    /SAVE THE NEW HIGH ADDRESS OF THE TEXT
03 A8 14          HIADD   /BUFFER ONCE THE INSERTION HAS BEEN
03 A9 18          0       /PERFORMED
03 AA D1          POPD    /GET THE PREVIOUS "HIADD"
03 AB 1A  OPENIT, LDAXD   /GET FROM THE PREVIOUS "HIADD"
03 AC 77          MOVMA   /MOVE TO THE NEW "HIADD"
```

```
03 AD 2B          DCXH     /DECREMENT BOTH MEMORY
03 AE 1B          DCXD     /ADDRESSES
03 AF 0B          DCXB     /AND THE WORD COUNT.
03 B0 78          MOVAB
03 B1 B1          ORAC
03 B2 C2          JNZ      /WE HAVE NOT MADE ENOUGH ROOM IN
03 B3 AB          OPENIT   /THE TEXT BUFFER FOR THE LINE OF
03 B4 03          0        /TEXT, SO OPEN IT UP SOME MORE.
03 B5 13  INSIT,  INXD     /INCREMENT TO THE POINT OF INSERTION
03 B6 E1          POPH     /GET THE BUFFER ADDRESS
03 B7 EB          XCHG
03 B8 1A  INSIT1, LDAXD    /GET A BUFFER CHARACTER
03 B9 FE          CPI      /INSERTED THE ENTIRE LINE?
03 BA 00          STERM
03 BB C8          RZ      ./YES, THEN DONE
03 BC 77          MOVMA    /NO, THEN SAVE THE CHAR. IN THE MAIN
03 BD 23          INXH     /TEXT BUFFER AND INCREMENT THE
03 BE 13          INXD     /TWO MEMORY ADDRESSES
03 BF C3          JMP      /THEN GET ANOTHER CHARACTER
03 C0 B8          INSIT1
03 C1 03          0
```

```
            /THIS SUBROUTINE IS USED TO INPUT THE LINE
            /NUMBER(S) THAT THE USER WANTS TO OPERATE ON.

03 C2 CD  INPUT,  CALL     /SET ALL OF THE REGISTERS TO ZERO
03 C3 29          CLNALL
03 C4 04          0
03 C5 26          MVIH     /LOAD H WITH 377 (USED TO DETERMINE
03 C6 FF          377      /IF A NUMBER HAS BEEN ENTERED)
03 C7 CD  INPUT1, CALL     /GET A CHARACTER FROM THE KEYBOARD
03 C8 06          IOECHO
03 C9 18          0
03 CA FE          CPI      /WAS THE CARRIAGE RETURN KEY PRESSED?
03 CB 0D          CR
03 CC CA          JZ       /YES, IT IS THE LINE TERMINATOR,
03 CD 01          TERM     /SO SEE IF NO LINE NUMBERS, ONE
03 CE 04          0        /LINE NUMBER OR TWO NUMBERS WERE ENTERED.
03 CF FE          CPI      /WAS THE DASH (-) KEY PRESSED, TO
03 D0 2D          055      /SEPARATE TWO LINE NUMBERS?
03 D1 CA          JZ       /YES, THEN SAVE THE NUMBER IN
03 D2 F3          SECT     /H&L IN "INITLN"
03 D3 03          0
03 D4 FE          CPI      /WAS A VALUE GREATER THAN ASCII 9
03 D5 3A          072      /ENTERED (OCTAL 071)
03 D6 D2          JNC      /YES, PRINT A ? AND RETURN TO THE
03 D7 EF          NOGOOD   /COMMAND DECODER
03 D8 03          0
03 D9 FE          CPI      /WAS A VALUE LESS THAN ASCII 0
03 DA 30          060      /ENTERED (OCTAL 060)
03 DB DA          JC       /YES, IGNORE IT.
03 DC EF          NOGOOD
03 DD 03          0
03 DE D6          SUI      /IT IS A VALID NUMBER, SO SUBTRACT
03 DF 30          060      /060 FROM IT.
03 E0 4F          MOVCA    /SAVE THE NUMBER IN C.
03 E1 06          MVIB     /SET B AND H TO ZERO
03 E2 00          000
03 E3 26          MVIH
03 E4 00          000
03 E5 E5  INOK,   PUSHH    /PUT THE TEMPORARY RESULT ON THE STACK
03 E6 D1          POPD     /AND GET IT BACK IN D&E
03 E7 29          DADH     /MULTIPLY BY 2
03 E8 29          DADH     /MULTIPLY BY 2 AGAIN (TOTAL = X4)
03 E9 19          DADD     /ADD, OR MULTIPLY BY 1 (TOTAL = X5)
03 EA 29          DADH     /MULTIPLY BY 2 AGAIN (TOTAL = X10)
03 EB 09          DADB     /NOW ADD IN THE NUMBER JUST TYPED IN.
03 EC C3          JMP      /THEN GET ANOTHER CHARACTER
03 ED C7          INPUT1   /FROM THE KEYBOARD
03 EE 03          0

03 EF E1  NOGOOD, POPH     /DESTROY THE RETURN ADDRESS
03 F0 C3          JMP      /THEN PRINT ?, C= BECAUSE THE
03 F1 6B          QUEST    /NUMBER OR COMMAND IS INVALID
03 F2 00          0
```

```
            /"SECT" IS EXECUTED IF A DASH (-) IS ENTERED

03 F3 7C  SECT,   MOVAH    /WAS THE FIRST OF 2 #'S
03 F4 B5          ORAL     /A 0 ? IF SO, ERROR.
03 F5 CA          JZ
03 F6 EF          NOGOOD
03 F7 03          0
03 F8 22  NMBOK,  SHLD     /SAVE THE INITIAL LINE NUMBER
03 F9 17          INITLN
03 FA 18          0
03 FB CD          CALL     /SET ALL OF THE REGISTERS TO ZERO
03 FC 29          CLNALL
03 FD 04          0
03 FE C3          JMP      /AND GET THE SECOND OF TWO NUMBERS
03 FF C7          INPUT1
04 00 03          0

            /"TERM" IS EXECUTED WHEN A CARRIAGE RETURN (CR)
            /IS ENTERED.

04 01 CD  TERM,   CALL     /A CARRIAGE RETURN WAS ENTERED, SO
04 02 39          CRLF     /PRINT A CARRIAGE RETURN (CR) AND A
04 03 02          0        /LINE FEED ON THE TTY OR CRT.
04 04 7C          MOVAH
04 05 FE          CPI      /ANY #'S TYPED IN AT ALL?
04 06 FF          377
04 07 CA          JZ       /APPARENTLY NOT.  THE USER WANTS TO
04 08 21          EVERY    /OPERATE ON THE ENTIRE TEXT BUFFER
04 09 04          0
04 0A B5          ORAL     /WAS A 0-000 TYPED IN ?
04 0B CA          JZ       /YES IT WAS, NO SUCH
04 0C EF          NOGOOD   /#!
04 0D 03          0
04 0E 3A  BLKOR1, LDA      /LOAD THE A REGISTER WITH THE MSBY
04 0F 18          SENSE    /OF THE 16-BIT "INITLN" (SEE THE
04 10 18          0        /FIRST PAGE OF THE LISTING)
04 11 FE          CPI      /IS IT STILL EQUAL TO THE PREVIOUSLY
04 12 C0          300      /INITIALIZED VALUE (SEE "GETCHR"
04 13 CA          JZ       /AFTER "QUEST1").  IF SO,
04 14 1C          ONEIN    /ONLY ONE LINE # WAS ENTERED, AND IT
04 15 04          0        /IS IN REGISTER PAIR H
04 16 22          SHLD     /TWO NUMBERS WERE ENTERED, SO SAVE
04 17 19          FINLN    /THE SECOND 16-BIT BINARY NUMBER
04 18 18          0        /IN "FINLN"
04 19 3E          MVIA     /THEN SET THE A REGISTER TO ONE TO
04 1A 01          001      /INDICATE THAT 2 #S WERE ENTERED.
04 1B C9          RET

04 1C 22  ONEIN,  SHLD     /ONE # WAS ENTERED BEFORE THE CR,
04 1D 17          INITLN   /SO SAVE IT IN "INITLN"
04 1E 18          0
04 1F AF          XRAA     /SET THE A REGISTER TO ZERO
04 20 C9          RET      /AND RETURN
```

```
04 21 3E  EVERY,  MVIA    /NO LINE NUMBERS WERE ENTERED, SO
04 22 02          002     /SET A TO TWO AND RETURN.
04 23 C9          RET

          /CONVERT THE BINARY VALUE BETWEEN 0 AND 9 TO ASCII
          /AND PRINT THE VALUE ON THE TTY OR CRT.

04 24 C6  BCDOUT, ADI
04 25 B0          260
04 26 C3          JMP
04 27 0F          100
04 28 18          0

04 29 AF  CLNALL, XRAA    /SET ALL OF THE REGISTERS TO ZERO.
04 2A 67          MOVHA
04 2B 6F          MOVLA
04 2C AF  CLNAE,  XRAA
04 2D 5F          MOVEA
04 2E 57          MOVDA
04 2F 4F          MOVCA
04 30 47          MOVBA
04 31 C9          RET

          /SEEK IS USED TO FIND THE FIRST CHARACTER IN THE LINE IN
          /THE TEXT BUFFER SPECIFIED BY THE LINE NUMBER
          /STORED IN "INITLN" (PUT THERE BY A CALL TO "INPUT").

04 32 2A  SEEK,   LHLD    /LOAD H&L WITH THE LINE NUMBER OF THE
04 33 17          INITLN  /LINE TO BE FOUND IN THE TEXT BUFFER
04 34 18          0
04 35 EB          XCHG    /MOVE THE NUMBER TO D&E
04 36 2A          LHLD    /LOAD H&L WITH THE STARTING ADDRESS
04 37 22          TXTBUF  /OF THE TEXT BUFFER.
04 38 18          0
04 39 1B  NXTSK,  DCXD    /D&E ARE THE INITIAL LINE
04 3A 7A          MOVAD
04 3B B3          ORAE    /ARE D&E 0 YET ?
04 3C C8          RZ      /YES, H&L ADDRESS THE LINE OF TEXT
                          /THAT WE WERE LOOKING FOR.
04 3D 7E  MORCHR, MOVAM   /D&E ARE NOT ZERO, GET A CHAR.
04 3E 23          INXH
04 3F FE          CPI     /FOUND THE END OF THE TEXT BUFFER?
04 40 9B          BTERM
04 41 CA          JZ      /YES, THEN WE CAN NOT GO ANY FURTHER.
04 42 EF          NOGOOD
04 43 03          0
04 44 FE          CPI     /FOUND A CARRIAGE RETURN IN THE
04 45 0D          CR      /TEXT BUFFER?
04 46 C2          JNZ     /NO, THEN KEEP LOOKING FOR EITHER
04 47 3D          MORCHR  /THE BUFFER TERMINATOR OR A CAR-
04 48 04          0       /RIAGE RETURN.
04 49 CD          CALL    /FOUND A CR, INCREMENT THE BCD
04 4A 5D          BFLINC  /LINE NUMBER BY ONE.
04 4B 04          0
```

```
04 4C C3          JMP     /THEN DECREMENT THE LINE COUNT IN
04 4D 39          NXTSK   /REGISTER PAIR D AND SEE IF IT IS
04 4E 04          0       /ZERO

          /"SEEK1" IS USED TO CONTINUE THE SEARCH, STARTING
          /WHERE "SEEK" LEFT OFF, UNTIL THE NUMBER OF LINES
          /SPECIFED BY "ADIFF" (USUALLY "FINLN" - "INITLN"
          /+1, BUT SOMETIMES "ADIFF" IS SET TO 000 001) ARE
          /PASSED.

04 4F E5  SEEK1,  PUSHH   /SAVE THE ADDRESS FROM "SEEK"
04 50 C1          POPB    /IN REGISTER PAIR B
04 51 EB          XCHG    /PUT THE ADDRESS IN D&E ALSO
04 52 2A          LHLD    /THEN GET THE "LINE COUNT" FROM
04 53 1B          ADIFF   /"ADIFF" (THE NUMBER OF LINES
04 54 18          0       /TO BE SKIPPED)
04 55 EB          XCHG    /ADDRESS IN H&L, "ADIFF" IN D&E
04 56 CD          CALL    /NOW SKIP THE NUMBER OF LINES
04 57 3D          MORCHR  /SPECIFIED BY THE NUMBER IN
04 58 04          0       /REGISTER PAIR D.
04 59 EB          XCHG    /FINAL ADDRESS NOW IS IN D&E
04 5A C5          PUSHB   /MOVE THE INITIAL ADDRESS DETERMINED
04 5B E1          POPH    /BY "SEEK" TO REGISTER PAIR H
04 5C C9          RET     /RETURN WITH ADDRESSES IN D&E AND H&L

          /THE BFLINC, BF1 AND BFOUT SUBROUTINES OPERATE ON THE
          /BCD LINE NUMBER STORED IN FOUR MEMORY LOCATIONS
          /(ONE DIGIT PER LOCATION) STARTING AT "BFUNIT" TO
          /"BFTHOU".  THESE LINE NUMBERS ARE PRINTED
          /DURING A LIST OPERATION.

          /INCREMENT THE BCD LINE NUMBER BY ONE

04 5D E5  BFLINC, PUSHH
04 5E C5          PUSHB
04 5F 21          LXIH
04 60 4A          BFUNIT
04 61 18          0
04 62 CD          CALL
04 63 F7          LINC1
04 64 05          0
04 65 C1          POPB
04 66 E1          POPH
04 67 C9          RET

          /SET THE BCD LINE NUMBER TO 0001

04 68 E5  BF1,    PUSHH
04 69 C5          PUSHB
04 6A 21          LXIH
04 6B 4A          BFUNIT
04 6C 18          0
04 6D CD          CALL
04 6E 0B          LINE12
```

```
04 6F 06            0
04 70 C1            POPB
04 71 E1            POPH
04 72 C9            RET

            /PRINT THE BCD LINE NUMBER

04 73 E5   BFOUT,   PUSHH
04 74 C5            PUSHB
04 75 21            LXIH
04 76 4D            BFTHOU
04 77 18            0
04 78 CD            CALL
04 79 1A            LNNMB1
04 7A 06            0
04 7B 21   BFOUT1,  LXIH
04 7C 39            EQU1
04 7D 12            0
04 7E CD            CALL
04 7F 7A            TOUT
04 80 05            0
04 81 C1            POPB
04 82 E1            POPH
04 83 C9            RET
```

```
          /IF THE S (SEARCH) COMMAND IS ENTERED, COME TO "FINDIT"

04 84 3E  FINDIT, MVIA    /PRINT AN EQUAL SIGN AFTER
04 85 3D          "="     /THE USER ENTERED THE S COMMAND.
04 86 CD          CALL
04 87 0F          IOO
04 88 18          0
04 89 CD          CALL    /SET THE BCD LINE NUMBERS USED ONLY
04 8A 08          LINEI   /BY THE S AND N COMMANDS TO 0001
04 8B 06          0
04 8C 21          LXIH    /LOAD REGISTER PAIR H WITH THE
04 8D 4E          STRING  /R/W MEMORY ADDRESS WHERE THE
04 8E 18          0       /USER'S STRING WILL BE STORED
04 8F 16          MVID    /LOAD THE D REGISTER WITH THE STRING
04 90 13          023     /TERMINATION CHARACTER (CTRL/S)
04 91 CD          CALL    /INPUT THE STRING TO BE SEARCHED
04 92 C8          STRIN   /FOR AND SAVE IT IN "STRING".  THE
04 93 04          0       /STRING IS TERMINATED WITH A CTRL/S.
04 94 3E          MVIA    /LOAD THE A REGISTER WITH 377 AND
04 95 FF          377     /SAVE IT IN "STRFND".  THIS IS A
04 96 32          STA     /"SENSE" MEMORY LOCATION.  IF A STRING
04 97 1F          STRFND  /IS FOUND THIS WILL BE SET TO ZERO,
04 98 18          0       /IF NO STRING IS FOUND, IT WILL BE 377.
04 99 2A          LHLD    /LOAD REGISTER PAIR H WITH THE
04 9A 22          TXTBUF  /STARTING ADDRESS OF THE TEXT BUFFER
04 9B 18          0
04 9C 22          SHLD    /SAVE THIS ADDRESS IN "SRCADD"
04 9D 1D          SRCADD  /FOR THE CONTINUE-SERACH (N)
04 9E 18          0       /COMMAND.
04 9F EB          XCHG    /MOVE THE ADDRESS TO D&E
04 A0 CD  NXTFND, CALL    /PERFORM THE SEARCH.  IF A MATCHING
04 A1 E8          NEWLIN  /STRING IS FOUND RETURN.  IF NOT
04 A2 04          0       /PRINT A ? AND GO TO THE COMMAND MODE
04 A3 21  CONS1,  LXIH    /A STRING WAS FOUND, SO SET "STRFND"
04 A4 1F          STRFND  /TO ZERO.  ALSO SET "EXOK" TO ZERO.
04 A5 18          0       /THIS IS USED BY THE EXCHANGE CMD.
04 A6 AF          XRAA
04 A7 77          MOVMA   /SET "STRFND" TO 0 FOR THE SEARCH AND NEXT
04 A8 23          INXH    /COMMANDS AND THEN SET THE NEXT LOCATION
04 A9 77          MOVMA   /TO 0 FOR THE EXCHANGE COMMAND
04 AA CD          CALL    /SET "ADIFF" TO 000 001, FOR THE
04 AB 63          DIFFI   /LISTING OF THE STRING
04 AC 03          0
04 AD 2A          LHLD    /SET H&L TO THE BEGINNING OF THE LINE
04 AE 1D          SRCADD  /(THIS IS SET BY THE "NEWLIN"
04 AF 18          0       /SUBROUTINE)
04 B0 3E          MVIA    /SET "PORL" TO L, SO THE LINE THAT
04 B1 4C          "L"     /CONTAINS THE MATCHING STRING CAN
04 B2 32          STA     /BE LISTED.
04 B3 41          PORL
04 B4 18          0
04 B5 CD          CALL    /PRINT A CARRIAGE RETURN AND LINE FEED.
04 B6 39          CRLF
04 B7 02          0
```

```
04 B8 E5          PUSHH
04 B9 C5          PUSHB
04 BA CD          CALL     /PRINT THE LINE NUMBER WHERE THE
04 BB 17          LNNMB    /MATCH OCCURRED.
04 BC 06          0
04 BD 21          LXIH     /PRINT TWO SPACES
04 BE 39          EQU1
04 BF 12          0
04 C0 CD          CALL
04 C1 7A          TOUT
04 C2 05          0
04 C3 C1          POPB
04 C4 E1          POPH
04 C5 C3          JMP      /AND THEN PRINT THE ENTIRE LINE
04 C6 D4          SCHAR    /OF TEXT WHERE THE MATCH OCCURRED.
04 C7 02          0

          /"STRIN" IS USED TO INPUT A STRING AND SAVE IT IN THE
          /MEMORY LOCATIONS ADDRESSED BY REGISTER PAIR H.
          /THE STRING IS TERMINATED WHEN THE ASCII VALUE
          /ENTERED EQUALS THE CONTENT OF THE D REGISTER.
          /"STRIN" MUST ONLY BE CALLED WITH AN ADDRESS IN
          /REGISTER PAIR H AND THE TERMINATION CHARACTER
          /IN THE D REGISTER.  A MAXIMUM OF 15 CHARACTERS
          /CAN BE ENTERED.

04 C8 0E  STRIN,  MVIC     /LOAD THE C REGISTER WITH THE MAXIMUM
04 C9 10          020      /NUMBER OF STRING CHARACTERS THAT CAN
04 CA CD  NSCIN,  CALL     /BE ENTERED; GET A STRING CHARACTER
04 CB 06          IOECHO
04 CC 18          0
04 CD FE          CPI
04 CE 03          CTRLC    /CTRL-C GETS US OUT OF HERE
04 CF CA          JZ       /BACK TO THE COMMAND MODE.
04 D0 6B          QUEST
04 D1 00          0
04 D2 BA          CMPD     /STRING TERMINATION CHARACTER?
04 D3 CA          JZ       /YES, THEN RETURN FROM "STRIN"
04 D4 DC          ENDLST
04 D5 04          0
04 D6 77          MOVMA    /NEITHER, SAVE THE CHARACTER.
04 D7 23          INXH     /INCREMENT THE MEMORY ADDRESS
04 D8 0D          DCRC     /DECREMENT THE STRING CHARACTER COUNTER
04 D9 C2          JNZ      /GET ANOTHER IF THE COUNT IS NON-ZERO
04 DA CA          NSCIN
04 DB 04          0
04 DC 36  ENDLST, MVIM     /END OF THE STRING, SAVE A TERMINATOR
04 DD 00          STERM
04 DE C9          RET

          /THIS SUBROUTINE ACTUALLY SEARCHES THE TEXT BUFFER
          /FOR A MATCHING STRING.  THIS IS USED BY THE S
          /AND N COMMANDS.
```

```
04 DF 13  ENTEST, INXD    /GET PAST THE CR IN THE TEXT BUFFER
04 E0 EB          XCHG
04 E1 22          SHLD    /SAVE THIS ADDRESS FOR THE
04 E2 1D          SRCADD  /"NEXT" COMMAND (N).
04 E3 18          0
04 E4 EB  NENTRY, XCHG    /PUT THE ADDRESS BACK INTO D&E
04 E5 CD          CALL
04 E6 F4          LINC    /INCREMENT THE DECIMAL (BCD) LINE
04 E7 05          0       /NUMBER
04 E8 21  NEWLIN, LXIH    /LOAD H&L WITH THE ADDRESS WHERE
04 E9 4E          STRING  /THE USER SUPPLIED STRING IS STORED
04 EA 18          0
04 EB 1A  NXTCHK, LDAXD   /GET A CHARACTER FROM THE TEXT BUFFER
04 EC FE          CPI     /IS IT A CARRIAGE RETURN?
04 ED 0D          CR
04 EE CA          JZ      /YES, THEN START THE SEARCH AGAIN
04 EF DF          ENTEST  /IN THE NEXT LINE
04 F0 04          0
04 F1 FE          CPI     /HAS THE END OF THE TEXT BUFFER
04 F2 9B          BTERM   /BEEN FOUND?
04 F3 C2          JNZ     /NO, THEN COMPARE THE TEXT BUFFER
04 F4 01          NOTERM  /CHARACTER TO THE FIRST STRING
04 F5 05          0       /CHARACTER.
04 F6 3A          LDA     /WE FOUND THE END OF THE TEXT BUFFER.
04 F7 1F          STRFND  /WERE ANY MATCHES FOUND (AS INDICATED
04 F8 18          0       /BY THE CONTENT OF "STRFND") ?
04 F9 FE          CPI
04 FA FF          377
04 FB CA          JZ      /NO MATCHES WERE FOUND, SO PRINT
04 FC EF          NOGOOD  /A ? AND THEN RETURN TO THE COM-
04 FD 03          0       /MAND MODE.
04 FE C3          JMP     /AT LEAST ONE MATCH OCCURRED, SO
04 FF 70          QUEST1  /SIMPLY RETURN TO THE COMMAND
05 00 00          0       /MODE

05 01 BE  NOTERM, CMPM    /NOT A CR OR BTERM, COMPARE THE TEXT
05 02 CA          JZ      /BUFFER CHARACTER TO THE USER'S STRING.
05 03 09          ONEOK   /IF THEY ARE EQUAL, TRY MATCHING SOME
05 04 05          0       /OTHER CHARACTERS.
05 05 13          INXD    /NO MATCH, INCREMENT THE BUFFER ADDR.
05 06 C3          JMP     /AND TRY FOR ANOTHER MATCH.
05 07 EB          NXTCHK
05 08 04          0

05 09 EB  ONEOK,  XCHG    /A SINGLE CHARACTER MATCH OCCURRED,
05 0A 22          SHLD    /SO SAVE THE ADDRESS WHERE THE
05 0B 34          ADDCNT  /FIRST MATCH OCCURRED
05 0C 18          0
05 0D EB          XCHG
05 0E 13  SMNXT,  INXD    /NOW TRY TO MATCH ALL OF THE RE-
05 0F 23          INXH    /MAINING CHARACTERS IN THE STRING
05 10 1A          LDAXD   /GET A TEXT BUFFER CHARACTER
05 11 BE          CMPM    /COMPARE THE STRING CHARACTER
05 12 CA          JZ      /THEY ARE EQUAL, KEEP PERFORMING
```

```
05 13 0E          SMNXT   /COMPARISONS UNTIL THEY ARE NOT
05 14 05          0       /EQUAL.
05 15 7E          MOVAM   /NO LONGER EQUAL, IS IT BECAUSE
05 16 FE          CPI     /THE TEXT BUFFER CHARACTER WAS
05 17 00          STERM   /COMPARED TO THE STRING TERMINATOR?
05 18 C2          JNZ     /NO, IT WAS NOT THE STRING TER-
05 19 E8          NEWLIN  /MINATOR, THE STRINGS JUST DO
05 1A 04          0       /NOT MATCH.
05 1B EB          XCHG    /FOUND THE STRING TERMINATOR
05 1C 22          SHLD    /THEY MATCHED "COMPLETELY" SO SAVE
05 1D 36          MNEVAL  /THE ADDRESS OF THE END OF THE STRING
05 1E 18          0       /IN THE TEXT BUFFER (FOR EXCHANGING)
05 1F C9          RET     /RETURN, A MATCH OCCURRED.

          /IF THE N (NEXT) COMMAND IS ENTERED, THE 8080 HAS TO
          /CONTINUE THE SEARCH, STARTING WERE IT LEFT OFF
          /FROM THE S (SEARCH) COMMAND.  WHEN THE N (NEXT)
          /COMMAND IS ENTERED, "CONSER" (CONTINUE SEARCHING)
          /IS EXECUTED.  HOWEVER, BEFORE THE SEARCH IS CONTINUED,
          /WE HAVE TO MAKE SURE THAT IT WAS STARTED!

05 20 21  CONSER, LXIH    /LOAD H&L WITH THE MEMORY ADDRESS
05 21 1F          STRFND  /WHERE THE SEARCH SENSE REGISTER
05 22 18          0       /IS STORED.
05 23 3E          MVIA    /LOAD THE A REGISTER WITH "NO
05 24 FF          377     /MATCHES OCCURRED"
05 25 BE          CMPM    /COMPARE IT TO "STRFND"
05 26 77          MOVMA   /SET "STRFND" TO 377 ANYWAY
05 27 CA          JZ      /IF NO MATCHES OCCURRED WITH THE S
05 28 6B          QUEST   /(SEARCH) COMMAND OR THE N COMMAND
05 29 00          0       /ENTERED, WE CAN NOT CONTINUE
05 2A 2A          LHLD    /A MATCH DID OCCUR THE LAST TIME
05 2B 1D          SRCADD  /THE S (SEARCH) CMD WAS USED, GET
05 2C 18          0       /THE ADDRESS WHERE WE SHOULD CONTINUE
05 2D 7E  FCR,    MOVAM   /NOW ADVANCE THE ADDRESS UNTIL THE
05 2E 23          INXH    /BEGINNING OF THE NEXT LINE OF TEXT
05 2F FE          CPI     /IN THE TEXT BUFFER IS FOUND.  THE
05 30 0D          CR      /SEARCH CAN THEN CONTINUE WITH THIS
05 31 C2          JNZ     /LINE OF TEXT.
05 32 2D          FCR
05 33 05          0
05 34 22          SHLD    /FOUND THE BEGINNING OF THE NEXT
05 35 1D          SRCADD  /LINE OF TEXT, SAVE ITS ADDRESS.
05 36 18          0
05 37 CD          CALL    /INCREMENT THE LINE NUMBER
05 38 E4          NENTRY  /AND THEN SEARCH FOR THE
05 39 04          0       /NEXT OCCURRENCE OF THE SAME STRING
05 3A C3          JMP     /(THE STRING BEING SEARCHED FOR
05 3B A3          CONS1   /IS STILL IN "STRING")
05 3C 04          0
```

```
            /"TAPE1" IS USED WHEN A SOURCE PAPER TAPE MUST BE
            /PUNCHED DUE TO THE P (PUNCH) COMMAND OF THE EDITOR.

05 3D 7E    TAPE1,  MOVAM   /GET A CHARACTER FROM THE TEXT BUFFER
05 3E FE            CPI     /IS IT A CARRIAGE RETURN?
05 3F 0D            CR
05 40 CA            JZ      /YES, THEN PUNCH A CARRIAGE RETURN
05 41 4F            OKCR    /AND A LINE FEED AND DECREMENT THE
05 42 05            0       /LINE COUNT
05 43 FE            CPI     /IS IT THE TEXT BUFFER TERMINATION
05 44 9B            BTERM   /CHARACTER?
05 45 CA            JZ      /YES, THEN PUNCH TRAILER BEFORE
05 46 15            TRLR    /RETURNING TO THE COMMAND MODE
05 47 03            0
05 48 CD    COUT,   CALL    /NOT A CR OR BTERM, SO PUNCH IT
05 49 09            IOP     /ON THE I/O PUNCH DEVICE
05 4A 18            0
05 4B 23    NXTLOC, INXH    /INCREMENT THE TEXT BUFFER ADDRESS
05 4C C3            JMP     /THEN GET ANOTHER CHARACTER AND
05 4D 3D            TAPE1   /POSSIBLY PUNCH IT.
05 4E 05            0

05 4F CD    OKCR,   CALL    /READ A CR FROM THE TEXT BUFFER, SO
05 50 61            PCRLF   /PUNCH A CARRIAGE RETURN AND A LINE
05 51 05            0       /FEED ON THE PAPER TAPE.
05 52 C3            JMP     /THEN DECREMENT THE LINE COUNT
05 53 FC            ENDPL   /IN A SECTION OF THE "LIST"
05 54 02            0       /INSTRUCTIONS.

            /"LDR" PUNCHES 100 CHARACTERS (OCTAL 200, HEX 80) ON
            /PAPER TAPE.  "LDR" IS CALLED BOTH WHEN A SOURCE AND
            /AN OBJECT VERSION OF THE PROGRAM IS PUNCHED.

05 55 0E    LDR,    MVIC    /LOAD THE C REGISTER WITH DECIMAL 100
05 56 64            144
05 57 3E    LDR1,   MVIA    /LOAD THE A REG. WITH THE VALUE TO
05 58 80            200     /BE PUNCHED (OCTAL 200, HEX 80)
05 59 CD            CALL    /PUNCH THE CONTENT OF THE A
05 5A 09            IOP     /REGISTER ON THE I/O PUNCH
05 5B 18            0       /DEVICE.
05 5C 0D            DCRC    /DECREMENT THE COUNT
05 5D C2            JNZ     /IF THE COUNT IS NON-ZERO, PUNCH
05 5E 57            LDR1    /THE SAME CHARACTER AGAIN
05 5F 05            0
05 60 C9            RET

05 61 3E    PCRLF,  MVIA    /PUNCH A CARRAGE RETURN AND A
05 62 8D            215     /LINE FEED ON THE PUNCH DEVICE.
05 63 CD            CALL
05 64 09            IOP
05 65 18            0
05 66 3E            MVIA
05 67 8A            212
05 68 C3            JMP
```

```
05 69 09          IOP
05 6A 18          0

          /IF THE C (CHANGE) COMMAND IS ENTERED, THE INSTRUC-
          /TIONS AT "CHANGE" ARE EXECUTED.  THIS MEANS THAT
          /THE SPECIFIED LINE OF TEXT IS FIRST DELETED, AND
          /THEN THE "INSERT" INSTRUCTIONS ARE EXECUTED SO THAT
          /A NEW LINE OF TEXT CAN BE ADDED TO THE TEXT BUFFER..
05 6B CD  CHANGE, CALL    /GET THE LINE NUMBER OF THE LINE
05 6C C2          INPUT   /OF TEXT TO BE "CHANGED"
05 6D 03          0
05 6E CD          CALL    /IFFC RETURRNOR C 3-5 IS ENTERED,
05 6F 26          CHKIT   /PRINT A ? AND RETURN TO THE COM-
05 70 06          0       /MAND MODE.  ONLY ONE LINE CAN BE
05 71 CD          CALL    /CHANGED.  RETURN IF ONLY ONE LINE
05 72 63          DIFF1   /NUMBER WAS ENTERED.  IF SO, SET
05 73 03          0       /"ADIFF" TO 000 001.
05 74 CD          CALL    /DELETE THE LINE OF TEXT
05 75 4D          DELIT
05 76 03          0
05 77 C3          JMP     /THEN GEE A LINE OF TEXT TO IN-
05 78 71          CHNG1   /SERT AND INSERT IT.
05 79 03          0

          /"TOUT" IS A GENERAL-PURPOSE SUBROUTINE THAT IS USED
          /TO PRINT ASCII MESSAGE STORED IN MEMORY ON THE I/O
          /OUTPUT DEVICE.  "TOUT" IS CALLED WITH THE ADDRESS
          /OF THE MESSAGE IN REGISTER PAIR H.  A ZERO MUST
          /BE STORED IN MEMORY AT THE END OF THE MESSAGE.

05 7A 7E  TOUT,   MOVAM   /GET A CHARACTER FROM MEMORY
05 7B FE          CPI     /IS IT THE MESSAGE TERMINATOR?
05 7C 00          STERM
05 7D C8          RZ      /YES, THEN WE ARE DONE.
05 7E CD          CALL    /NOT A TERMINATOR, PRINT THE
05 7F 0F          IOO     /ASCII CHARACTER ON THE I/O
05 80 18          0       /OUTPUT DEVICE
05 81 23          INXH    /INCREMENT THE MEMORY ADDRESS
05 82 C3          JMP     /THEN GET ANOTHER CHARACTER AND
05 83 7A          TOUT    /SEE WHAT IT IS.
05 84 05          0

          /IF THE Q (QUERY) COMMAND IS ENTERED, WE HAVE TO
          /PRINT THE HIGHEST ADDRESS USED, THE HIGHEST AD-
          /DRESS AVAILABLE AND THE  NUMBER OF LINES IN THE
          /TEXT BUFFER.

05 85 CD  MEMOUT, CALL    /PRINT A CARRIAGE RETURN AND
05 86 39          CRLF    /A LINE FEED ON THE IOO DEVICE.
05 87 02          0
05 88 2A          LHLD    /SET H&L = TO THE HIGHEST ADDRESS
05 89 14          HIADD   /USED BY THE TEXT BUFFER.
05 8A 18          0
05 8B CD          CALL    /PRINT THIS ADDRESS
```

```
05 8C AD          HLOUT
05 8D 05          0
05 8E CD          CALL     /THEN TYPE A CR AND LF
05 8F 39          CRLF
05 90 02          0
05 91 2A          LHLD     /SET H&L = TO THE LAST R-W
05 92 24          ENDRAM   /MEMORY LOCATION THAT WAS
05 93 18          0        /ALLOCATED BY THE USER.
05 94 CD          CALL     /AND THEN TYPE IT OUT.
05 95 AD          HLOUT
05 96 05          0
05 97 CD          CALL     /COUNT THE NUMBER OF LINES
05 98 26          CNTLN    /IN THE TEXT BUFFER
05 99 01          0
05 9A CD  CAT,    CALL     /TYPE A CR, THEN AN LF
05 9B 39          CRLF
05 9C 02          0
05 9D 21          LXIH     /LOAD REGISTER PAIR H WITH THE
05 9E 30          LNER+4   /ADDRESS OF THE MESSAGE "LINE #"
05 9F 12          0
05 A0 CD          CALL     /PRINT THE MESSAGE "LINE #"
05 A1 7A          TOUT
05 A2 05          0
05 A3 23          INXH     /INCREMENT THE ADDRESS PAST THE 0
05 A4 CD          CALL     /THEN PRINT THE MESSAGE "=  "
05 A5 7A          TOUT     /WHICH IS STORED IN MEMORY JUST
05 A6 05          0        /AFTER "LINE #"
05 A7 CD          CALL
05 A8 73          BFOUT    /PRINT THE NUMBER OF LINES
05 A9 04          0
05 AA C3          JMP      /THEN GET ANOTHER COMMAND
05 AB 70          QUESTI
05 AC 00          0

          /THIS SUBROUTINE PRINTS THE CONTENT OF REGISTER PAIR
          /H ON THE IOO DEVICE AS EITHER OCTAL OR HEXADECIMAL
          /NUMBERS, BASED ON THE STATE OF THE ""HORO" MEMORY
          /LOCATION.  IF IT CONTAINS OCTAL 110 (HEX 48; ASCII
          /H) HEX IS USED, OTHERWISE OCTAL IS USED.

05 AD 7C  HLOUT,  MOVAH    /GET THE H REGISTER
05 AE CD          CALL     /PRINT IT AS EITHER OCTAL OR
05 AF B2          OCTOUT   /HEX ON THE IOO DEVICE.
05 B0 05          0
05 B1 7D          MOVAL    /THEN GET L AND PRINT IT
05 B2 4F  OCTOUT, MOVCA    /SAVE THE NUMBER IN REGISTER C
05 B3 3A          LDA      /ARE WE TO PRINT HEXADECIMAL OR
05 B4 3E          HORO     /OCTAL NUMBERS ?
05 B5 18          0
05 B6 FE          CPI
05 B7 48          "H"
05 B8 C2          JNZ      /"HORO" DOES NOT CONTAIN AN ASCII
05 B9 D8          OCTI     /H, SO THE NUMBERS WILL BE PRINTED
05 BA 05          0        /IN OCTAL
```

```
05 BB 79        MOVAC   /"HORO" CONTAINS AN H, PRINT HEX
05 BC E6        ANI     /SAVE THE FOUR MSBS
05 BD F0        360
05 BE 0F        RRC     /ROTATE THEM INTO THE FOUR LSBS
05 BF 0F        RRC
05 C0 0F        RRC
05 C1 0F        RRC
05 C2 CD        CALL    /CONVERT THE BINARY VALUE TO
05 C3 CE        PHEX    /ASCII-BASED HEXADECIMAL AND
05 C4 05        0       /THEN PRINT THE CHARACTER
05 C5 79        MOVAC   /GET THE SAME 8-BIT NUMBER AGAIN
05 C6 E6        ANI     /SAVE THE FOUR LSBS
05 C7 0F        017
05 C8 CD        CALL    /CONVERT THE BINARY VALUE TO
05 C9 CE        PHEX    /ASCII-BASED HEXADECIMAL AND
05 CA 05        0       /PRINT IT.
05 CB C3        JMP     /THEN PRINT A SPACE AFTER THE TWO-
05 CC EF        SPCAFT  /DIGIT HEXADECIMAL NUMBER.
05 CD 05        0
```

```
05 CE FE  PHEX,   CPI      /IS A 0-9 TO BE PRINTED?
05 CF 0A          012
05 D0 DA          JC       /YES, THEN ADD 260 TO THE CONTENT
05 D1 24          BCDOUT   /OF THE A REGISTER AND PRINT THE
05 D2 04          0        /RESULT.
05 D3 C6          ADI      /ITS A-F, ADD 267
05 D4 B7          267
05 D5 C3          JMP      /AND THEN PRINT IT.
05 D6 0F          100
05 D7 18          0

          /THIS SECTION OF OCTOUT IS USED TO PRINT THE CONTENT
          /OF THE A REGISTER IN THE FORM OF THREE OCTAL DIGITS

05 D8 79  OCT1,   MOVAC    /GET THE NUMBER INTO A.
05 D9 E6          ANI      /SAVE THE TWO MSBS IN THE A REG.
05 DA C0          300
05 DB 07          RLC      /ROTATE THEM INTO THE TWO LSBS
05 DC 07          RLC
05 DD CD          CALL     /ADD 260 TO THE RESULT AND PRINT
05 DE 24          BCDOUT   /IT ON THE I/O OUTPUT DEVICE
05 DF 04          0
05 E0 79          MOVAC    /GET THE SAME NUMBER INTO A AGAIN
05 E1 E6          ANI      /SAVE THE THREE "MIDDLE" BITS
05 E2 38          070
05 E3 0F          RRC      /ROTATE THEM INTO THE THREE LSBS
05 E4 0F          RRC
05 E5 0F          RRC
05 E6 CD          CALL     /ADD 260 TO THE RESULT AND PRINT
05 E7 24          BCDOUT   /IT ON THE I/O OUTPUT DEVICE.
05 E8 04          0
05 E9 79          MOVAC    /GET THE SAME NUMBER AGAIN
05 EA E6          ANI      /SAVE THE THREE LSBS
05 EB 07          007
05 EC CD          CALL     /ADD 260 TO THE RESULT AND PRINT
05 ED 24          BCDOUT   /IT ON THE I/O OUTPUT DEVICE.
05 EE 04          0
05 EF 3E  SPCAFT, MVIA     /PRINT A SPACE AFTER THE NUMBER
05 F0 A0          240
05 F1 C3          JMP
05 F2 0F          100
05 F3 18          0

          /THE LINC, LINE1 AND LNNMB SUBROUTINES ARE USED TO
          /GENERATE LINE NUMBERS USED WITH THE S (SEARCH) AND
          /N (NEXT) COMMANDS.  THEY ARE ALSO USED TO GENERATE
          /LINE NUMBERS FOR ERROR MESSAGES DURING THE AS-
          /SEMBLY PROCESS.

          /THIS SUBROUTINE INCREMENTS THE BCD LINE NUMBER BY 1.

05 F4 21  LINC,   LXIH
05 F5 28          UNIT
05 F6 18          0
```

```
05 F7 0E  LINC1,  MVIC
05 F8 04          004      /# OF DECIMAL DIGITS
05 F9 34  NXTNBM, INRM
05 FA 7E          MOVAM
05 FB FE          CPI
05 FC BA          272      /1 MORE THAN 9
05 FD D8          RC       /YES, 272 WAS BIGGER THAN THE #
05 FE 36          MVIM
05 FF B0          260
06 00 23          INXH
06 01 0D          DCRC
06 02 C2          JNZ
06 03 F9          NXTNBM
06 04 05          0
06 05 C3          JMP      /SOMETHINGS WRONG BECAUSE THE
06 06 EF          NOGOOD   /NUMBER IS GREATER THAN 4 DIGITS
06 07 03          0        /OR 9,999, SO POP THE RET AND ?, C=

          /THIS SUBROUTINE SETS THE BCD LINE NUMBER TO 0001.

06 08 21  LINE1,  LXIH     /THIS SETS THE LINE COUNTER TO 0001
06 09 28          UNIT
06 0A 18          0
06 0B 36  LINE12, MVIM     /SAVE AN ASCII 1 IN MEMORY
06 0C B1          261
06 0D 0E          MVIC     /THEN SAVE THREE ASCII 0'S IN THE
06 0E 03          003      /NEXT THREE CONSECUTIVE MEMORY LOCATIONS
06 0F 23  ZEROIT, INXH     /INCREMENT PAST THE ASCII 1
06 10 36          MVIM     /SAVE AN ASCII 0
06 11 B0          260
06 12 0D          DCRC     /THREE OF THEM SAVED YET ?
06 13 C2          JNZ      /NO, SAVE ANOTHER ASCII 0
06 14 0F          ZEROIT
06 15 06          0
06 16 C9          RET      /YES, WE SAVED 0001 IN MEMORY

          /THIS SUBROUTINE PRINTS THE FOUR-DIGIT BCD LINE NUMBER

06 17 21  LNNMB,  LXIH     /TYPE OUT THE 4 DIGIT
06 18 2B          THOU     /DECIMAL #.
06 19 18          0
06 1A 0E  LNNMB1, MVIC     /C=THE DIGIT COUNTER (CURRENTLY 4)
06 1B 04          004
06 1C 7E  NPRINT, MOVAM    /GET A BCD DIGIT FROM MEMORY (ASCII)
06 1D CD          CALL
06 1E 0F          IOO      /AND PRINT IT OUT
06 1F 18          0
06 20 2B          DCXH     /DECREMENT H&L TO A LESS SIGNIFICANT DIGIT
06 21 0D          DCRC     /DECREMENT THE DIGIT COUNTER
06 22 C8          RZ       /PRINTED ALL FOUR DIGITS, SO RETURN
06 23 C3          JMP      /HAVEN'T PRINTED 4 YET, KEEP GOING
06 24 1C          NPRINT
06 25 06          0
```

```
            /"CHKIT" IS CALLED BY SOME "COMMANDS" IF ONLY
            /ONE LINE NUMBER CAN BE SPECIFIED IN THE COMMAND
            /(INSERT AND CHANGE).  IF ONLY ONE LINE NUMBER
            /WAS ENTERED BY THE USER, CONTROL RETURNS FROM
            /THE SUBROUTINE.  OTHERWISE, CONTROL RETURNS TO
            /THE COMMAND DECODER (COMMAND MODE).

06 26 FE    CHKIT,  CPI     /A=001 OR 002 ?
06 27 01            001
06 28 D8            RC
06 29 C3            JMP
06 2A EF            NOGOOD
06 2B 03            0

            /"CLEAN" IS USED WHEN TEA IS STARTED AND NO SOURCE IS
            /CURRENTLY STORED IN MEMORY OR WHEN THE ENTIRE TEXT
            /BUFFER IS DELETED.

06 2C 2A    CLEAN,  LHLD    /GET THE USER ESTABLISHED "INITIAL
06 2D 22            TXTBUF  /ADDRESS" FOR THE TEXT BUFFER.
06 2E 18            0
06 2F 36            MVIM    /SAVE A BUFFER TERMINATOR AT THE
06 30 9B            BTERM   /BEGINNING OF THE TEXT BUFFER
06 31 22            SHLD    /SAVE THE ADDRESS AS THE
06 32 14            HIADD   /HIGHEST ADDRESS USED
06 33 18            0
06 34 22            SHLD    /ALSO INITIALIZE "SRCADD" WHICH
06 35 1D            SRCADD  /IS USED BY THE N (NEXT) COMMAND
06 36 18            0
06 37 22            SHLD    /ALSO INITIALIZE "TFCLIN" (TEXT- FIRST
06 38 2E            TFCLIN  /CHARACTER IN A LINE) FOR THE ASSEMBLER
06 39 18            0
06 3A 21            LXIH    /LOAD REGISTER PAIR H WITH THE R/W
06 3B A8            BUFF2   /MEMORY ADDRESS WHERE THE ONE-LINE
06 3C 18            0       /(MINI-BUFFER) IS.
06 3D 2B            DCXH    /DECREMENT THE ADDRESS
06 3E 36            MVIM    /AND SAVE A CR "JUST IN FRONT OF" THE
06 3F 0D            CR      /SINGLE LINE OF TEXT
06 40 2E            MVIL    /LOAD REGISTER PAIR H WITH 377 377
06 41 FF            377     /AND SAVE THE VALUE IN R/W MEMORY.
06 42 26            MVIH
06 43 FF            377
06 44 22            SHLD    /THIS "INDICATES" THAT NO MATCHES HAVE
06 45 1F            STRFND  /BEEN FOUND AND THAT THE EXCHANGE
06 46 18            0       /COMMAND CAN NOT BE USED
06 47 CD            CALL    /THEN SET ALL OF THE REGISTERS
06 48 29            CLNALL  /TO ZERO
06 49 04            0
06 4A 32            STA     /CLEAR THE READER-KEYBOARD SWITCH
06 4B 40            RORK    /TO THE KEYBOARD MODE
06 4C 18            0
06 4D 22            SHLD    /AND THEN SET "INITLN", TO ZERO
06 4E 17            INITLN
06 4F 18            0
```

```
06 50 C9            RET

          /THESE ARE THE GENERAL-PURPOSE I/O SUBROUTINES THAT
          /TEA ACCESSES THROUGH THE I/O JUMP TABLE.  THESE
          /SUBROUTINES CAN BE MODIFIED OR NEW SUBROUTINES
          /CAN BE ADDED AND ACCESSED THROUGH A MODIFIED JUMP
          /TABLE.

06 51 DB  INCHK,  IN        /JUST GET THE KEYBOARD FLAG
06 52 01          001
06 53 E6          ANI       /AND SET THE FLAGS BASED ON ITS STATE
06 54 01          001
06 55 C9          RET

06 56 D3  IORDR,  OUT       /PULSE THE READER CONTROL RELAY
06 57 01          001
06 58 00          0         /THERE IS ENOUGH ROOM HERE FOR ANY
06 59 00          0         /TIME DELAY SOFTWARE REQUIRED FOR A
06 5A 00          0         /SOFTWARE CONTROLLED READER-CONTROL
06 5B 00          0         /RELAY.
06 5C 00          0
06 5D 00          0
06 5E 00          0
06 5F 00          0
06 60 00          0
06 61 00          0
06 62 00          0
06 63 00          0
06 64 00          0
06 65 00          0
06 66 00          0
06 67 00          0
06 68 00          0
06 69 00          0
06 6A DB  TTYI,   IN        /GET THE PRINTER'S AND KEYBOARD'S
06 6B 01          001       /FLAGS
06 6C E6          ANI       /SAVE JUST THE KEYBOARD'S FLAG
06 6D 01          001
06 6E CA          JZ        /IF THE FLAG IS ZERO, WAIT FOR IT
06 6F 6A          TTYI      /TO BE A LOGIC ONE, MEANING THAT A
06 70 06          0         /KEY HAS BEEN PRESSED.
06 71 DB          IN        /A KEY WAS PRESSED, INPUT ITS
06 72 00          000       /ASCII VALUE.
06 73 E6          ANI       /MASK OUT ANY PARITY BIT
06 74 7F          177
06 75 C9          RET       /AND RETURN WITH IT IN THE A REGISTER

06 76 CD  TTYIN,  CALL      /GET A KEYBOARD CHARACTER
06 77 6A          TTYI      /AND PRINT IT.
06 78 06          0
06 79 47  TTYOUT, MOVBA     /SAVE THE CHARACTER IN B
06 7A DB          IN
06 7B 01          001
06 7C E6          ANI
```

```
06 7D 04        004     /CHECK TO SEE IF THE PRINTER
06 7E CA        JZ      /IS READY.  IT ISN'T, KEEP CHECKING
06 7F 7A        TTYOUT+1
06 80 06        0
06 81 78        MOVAB   /PRINTER IS READY, GET THE CHARACTER
06 82 D3        OUT
06 83 00        000     /AND PRINT IT
06 84 CD        CALL    /SEE IF ANY KEYS HAVE BEEN PRESSED
06 85 00        IOTEST
06 86 18        0
06 87 78        MOVAB   /GET THE PRINTER CHARACTER BACK IN A
06 88 C8        RZ      /NO KEYS PRESSED
06 89 CD        CALL    /A KEY WAS PRESSED, FIND OUT WHAT IT IS
06 8A 03        IONOEC
06 8B 18        0
06 8C FE        CPI
06 8D 03        CTRLC   /WAS A CTRL-C TYPED IN?
06 8E 78        MOVAB   /GET THE OLD PRINTER CHARACTER
06 8F C0        RNZ     /NO, NOT A CNTRL-C, SO IGNORE IT
06 90 31        LXISP   /IT WAS A CNTRL-C,
06 91 A0        STACK   /RESET THE STACK POINTER
06 92 18        0
06 93 C3        JMP     /AND SEE WHAT THE USER'S UP TO
06 94 6B        QUEST
06 95 00        0
```

```
          /"MMEXC" IS EXECUTED IF THERE IS NOT ENOUGH R/W
          /MEMORY FOR THE TEXT BUFFER OR THE ASSEMBLER'S SYM-
          /BOL TABLE AND DEFINED-BYTE TABLE COLLIDE.

06 96 21  MMEXC,  LXIH    /LOAD REGISTER PAIR H WITH THE AD-
06 97 F3          MEAIC   /DRESS FOR THE MEI ERROR MESSAGE.
06 98 11          0
06 99 CD          CALL    /PRINT THE MESSAGE
06 9A 7A          TOUT
06 9B 05          0
06 9C C3          JMP     /THEN ENTER THE COMMAND MODE,
06 9D 70          QUEST1  /REGARDLESS OF THE OPERATION
06 9E 00          0       /THAT CAUSED THE ERROR.

06 9F 21  EXCHG,  LXIH    /HAVE ANY STRINGS BEEN FOUND THAT WE
06 A0 1F          STRFND  /CAN EXCHANGE?
06 A1 18          0
06 A2 7E          MOVAM
06 A3 FE          CPI
06 A4 FF          377
06 A5 CA          JZ      /NO, NO STRINGS
06 A6 6B          QUEST
06 A7 00          0
06 A8 23          INXH    /YES, THERE WAS A MATCH
06 A9 3E          MVIA    /HAS AN EXCHANGE ALREADY
06 AA FF          377     /TAKEN PLACE?
06 AB BE          CMPM
06 AC 77          MOVMA
06 AD CA          JZ      /YES, THEN WE CAN NOT EXCHANGE AGAIN.
06 AE 6B          QUEST
06 AF 00          0
06 B0 CD          CALL    /A STRING WAS FOUND THAT HAS NOT BEEN
06 B1 06          IOECHO  /PREVIOUSLY EXCHANGED, SO GET A COM-
06 B2 18          0       /MAND
06 B3 FE          CPI     /WANT TO EXCHANGE THE SAME STRING?
06 B4 53          "S"
06 B5 CA          JZ      /YES, DELETE THE OLD ONE AND
06 B6 C5          DAIES   /ADD THE NEW ONE.
06 B7 06          0
06 B8 FE          CPI     /IS THE COMMAND E=?
06 B9 3D          "="
06 BA C2          JNZ     /NO, THEN IGNORE THE COMMAND
06 BB 6B          QUEST
06 BC 00          0
06 BD 21          LXIH    /YES, LOAD REGISTER PAIR H WITH
06 BE 6A          EXSTR   /R/W MEMORY ADDRESS WHERE THE
06 BF 18          0       /NEW STRING CAN BE STORED.
06 C0 16          MVID    /LOAD THE D REGISTER WITH THE
06 C1 05          005     /STRING TERMINATOR (CTRL-E)
06 C2 CD          CALL    /INPUT THE STRING FROM THE KEYBOARD
06 C3 C8          STRIN   /AND SAVE IT IN R/W MEMORY
06 C4 04          0
06 C5 2A  DAIES,  LHLD    /NOW DELETE THE OLD STRING
06 C6 34          ADDCNT  /GET ITS STARTING ADDRESS
```

```
06 C7 18            0        /(FROM THE SEARCH COMMAND)
06 C8 EB            XCHG
06 C9 2A            LHLD     /GET ITS FINAL ADDRESS
06 CA 36            MNEVAL   (FROM THE SEARCH COMMAND)
06 CB 18            0
06 CC 7E  DELSTR,   MOVAM    /GET A CHARACTER AFTER THE STRING
06 CD 12            STAXD    /MOVE IT DOWN ON TOP OF THE STRING
06 CE FE            CPI      /MOVED THE TEXT BUFFER TERMINATOR
06 CF 9B            BTERM    /YET?
06 D0 CA            JZ       /YES, THEN INSERT THE NEW STRING
06 D1 D8            INSSTR
06 D2 06            0
06 D3 23            INXH     /NO, MOVE ANOTHER CHARACTER ON
06 D4 13            INXD     /TOP OF THE OLD STRING
06 D5 C3            JMP
06 D6 CC            DELSTR
06 D7 06            0
06 D8 EB  INSSTR,   XCHG     /GET THE TEXT ADDRESS INTO H&L
06 D9 22            SHLD     /SAVE THE NEW HI ADDRESS
06 DA 14            HIADD
06 DB 18            0
06 DC 2A            LHLD     /GET THE ADDRESS WHERE THE
06 DD 34            ADDCNT   /NEW STRING IS TO BE INSERTED
06 DE 18            0
06 DF 22            SHLD     /SAVE IT IN "INITLN" FOR THE
06 E0 17            INITLN   /"STOBUF" (STORE BUFFER)
06 E1 18            0        /SUBROUTINE
06 E2 21            LXIH     /LOAD REGISTER PAIR H WITH THE
06 E3 64            EXSTR    /STARTING ADDRESS OF THE NEW
06 E4 18            0        /STRING (WHERE IT IS IN MEMORY)
06 E5 CD            CALL     /STORE THE STRING IN THE
06 E6 86            STOBUF   /TEXT BUFFER
06 E7 03            0
06 E8 C3            JMP      /THEN GET ANOTHER COMMAND
06 E9 70            QUEST1
06 EA 00            0

          /"ADRIN" IS CALLED AT THE BEGINNING OF THE PROGRAM
          /WHEN THE INITIAL AND FINAL R/W MEMORY ADDRESSES
          /MUST BE ENTERED.

06 EB CD  ADRIN,    CALL     /GET EITHER AN O, H OR CR.
06 EC 06            IOECHO   /FROM THE KEYBOARD.
06 ED 18            0
06 EE FE            CPI      /WANT THE SAME PREVIOUS ADDRESS?
06 EF 0D            CR
06 F0 C8            RZ       /YES, THE ADDRESS IS ALREADY IN D&E
06 F1 FE            CPI      /WANT TO ENTER A HEXADECIMAL ADDRESS?
06 F2 48            "H"
06 F3 CA            JZ       /YES, THEN GET TWO TWO-DIGIT
06 F4 22            TWOHEX   /HEXADECIMAL NUMBERS
06 F5 07            0
06 F6 FE            CPI      /WANT TO ENTER AN OCTAL ADDRESS?
06 F7 4F            "O"
```

```
06 F8 C2          JNZ     /NO, THEN IGNORE THE CHARACTER
06 F9 EB          ADRIN   /AND GET ANOTHER ONE.
06 FA 06          0
06 FB CD          CALL    /OK, TWO THREE-DIGIT OCTAL NUMBERS
06 FC FF          OCTIN   /WILL BE ENTERED.  GET ONE THREE-
06 FD 06          0       /DIGIT NUMBER INTO THE D REGISTER
06 FE 53          MOVDE   /SAVE THIS MSBY IN THE D REGISTER
06 FF AF  OCTIN,  XRAA    /SET THE A AND E REGISTERS TO ZERO
07 00 5F          MOVEA
07 01 0E          MVIC    /LOAD THE C REGISTER WITH THREE,
07 02 03          003     /BECAUSE THREE DIGITS WILL BE INPUT
07 03 CD  NXTOCT, CALL    /GET AN ASCII VALUE FROM THE KEYBOARD
07 04 06          IOECHO
07 05 18          0
07 06 FE  SPCOIN, CPI     /IS IT A VALID OCTAL DIGIT?
07 07 30          "0"
07 08 DA          JC      /NO, ITS VALUE IS LESS THAN AN
07 09 03          NXTOCT  /ASCII 0.
07 0A 07          0
07 0B FE          CPI     /ITS EQUAL TO OR GREATER THAN AN
07 0C 38          "8"     /ASCII 0, IS IT LESS THAN AN ASCII
07 0D D2          JNC     /8?  IF NOT, IGNORE IT.
07 0E 03          NXTOCT
07 0F 07          0
07 10 E6          ANI     /ITS A VALID OCTAL NUMBER, SAVE THE
07 11 07          007     /THREE LSBS.
07 12 47          MOVBA   /SAVE THE VALUE IN B
07 13 7B          MOVAE   /GET THE PREVIOUS VALUE
07 14 07          RLC     /MULTIPLY IT BY EIGHT
07 15 07          RLC
07 16 07          RLC
07 17 80          ADDB    /ADD THE VALUE JUST ENTERED.
07 18 5F          MOVEA   /SAVE THE RESULT IN E
07 19 0D          DCRC    /DECREMENT THE DIGIT COUNT
07 1A C2          JNZ     /IF THE DIGIT COUNT IS NON-ZERO,
07 1B 03          NXTOCT  /GET ANOTHER VALUE FROM THE KYBD.
07 1C 07          0
07 1D 3E  SPC,    MVIA    /PRINT A SPACE AFTER THE LAST DIGIT.
07 1E A0          240
07 1F C3          JMP
07 20 0F          100
07 21 18          0

          /"TWOHEX" IS CALLED IF THE USER ENTERED AN "H".  THIS
          /MEANS THAT THE ADDRESS MUST BE ENTERED USING HEX.

07 22 CD  TWOHEX, CALL    /GET THE FIRST TWO OF FOUR HEX-
07 23 26          HEXIN   /ADECIMAL DIGITS.
07 24 07          0
07 25 53          MOVDE   /SAVE THE MSBY IN THE D REGISTER.
07 26 AF  HEXIN,  XRAA    /SET THE A AND E REGISTERS TO ZERO
07 27 5F          MOVEA
07 28 0E          MVIC    /LOAD THE C REGISTER WITH 002,
07 29 02          002     /BECAUSE TWO DIGITS WILL BE ENTERED.
```

```
07 2A CD  NXTHEX, CALL    /GET A KEYBOARD CHARACTER.
07 2B 06          IOECHO
07 2C 18          0
07 2D FE  SKPHIN, CPI     /LESS THAN ASCII 0?
07 2E 30          "0"
07 2F DA          JC      /YES, THEN IGNORE IT.
07 30 2A          NXTHEX
07 31 07          0
07 32 FE          CPI     /LESS THAN ASCII :?
07 33 3A          ":"
07 34 DA          JC      /YES, THEN IT MUST BE ASCII 0 - 9.
07 35 43          HEX09
07 36 07          0
07 37 FE          CPI     /LESS THAN ASCII A?
07 38 41          "A"
07 39 DA          JC      /YES, THEN IGNORE IT.
07 3A 2A          NXTHEX
07 3B 07          0
07 3C FE          CPI     /EQUAL TO OR GREATER THAN ASCII G?
07 3D 47          "G"
07 3E D2          JNC     /YES, THEN IGNORE IT.
07 3F 2A          NXTHEX
07 40 07          0
07 41 C6          ADI     /ITS A - F, ADD 011 TO IT.
07 42 09          011
07 43 E6  HEX09,  ANI     /SAVE THE FOUR LSBS
07 44 0F          017
07 45 47          MOVBA   /SAVE THE VALUE IN THE B REGISTER
07 46 7B          MOVAE   /GET THE PREVIOUS VALUE
07 47 07          RLC     /MULTIPLY IT BY 16
07 48 07          RLC
07 49 07          RLC
07 4A 07          RLC
07 4B 80          ADDB    /ADD THE VALUE JUST ENTERED
07 4C 5F          MOVEA   /SAVE THE RESULT IN E.
07 4D 0D          DCRC    /DECREMENT THE DIGIT COUNT
07 4E C2          JNZ     /IF THE COUNT IS NON-ZERO,
07 4F 2A          NXTHEX  /ANOTHER CHARACTER MUST BE ENTERED.
07 50 07          0
07 51 C9          RET

07 52 16  BOOT,   MVID
07 53 00          000     /INITIALIZE THE CHECKSUM
07 54 CD  LDRIN,  CALL    /READ A CHARACTER FROM THE TAPE
07 55 0C          IOR
07 56 18          0
07 57 FE          CPI     /IS IT THE LEADER (200'S) ?
07 58 80          200
07 59 CA          JZ      /YES, KEEP READING UNTIL WE
07 5A 54          LDRIN   /PASS ALL THE LEADER
07 5B 07          0
07 5C FE  CHKFRM, CPI     /NO, NOT LEADER IS IT THE ADDRESS FLAG ?
07 5D 40          100
07 5E C2          JNZ     /NO, SEE IF IT WAS THE CHECKSUM FLAG (300)
```

```
07 5F 6F          NOTA
07 60 07          0
07 61 CD          CALL     /YES, IT WAS THE ADDRESS FLAG (A 100)
07 62 92          ADDCHK   /GET THE NEXT 2 FRAMES, WHICH ARE THE
07 63 07          0        /HIGH ADDRESS
07 64 67          MOVHA
07 65 CD          CALL     /THEN GET THE 2 FRAMES WHICH ARE
07 66 92          ADDCHK   /THE LO ADDRESS
07 67 07          0
07 68 6F          MOVLA
07 69 CD  BNXTIN, CALL     /NOW GET ANOTHER FRAME
07 6A 0C          IOR      /AND PROCESS IT
07 6B 18          0
07 6C C3          JMP
07 6D 5C          CHKFRM
07 6E 07          0
07 6F FE  NOTA,   CPI      /WAS NOT A 100 FOR AN ADDRESS
07 70 C0          300      /IS IT THE CHECKSUM FLAG (A 300) ?
07 71 C2          JNZ
07 72 84          NOTSUM   /NO, THEN TREAT IT AS DATA
07 73 07          0
07 74 CD          CALL     /IT WAS THE FLAG, GET THE
07 75 9A          BBYTE    /2 FOLLOWING FRAMES, BUT DON'T
07 76 07          0        /ADD THE CHECKSUM TO THEM.
07 77 BA          CMPD     /COMPARE THE CALCULATED VERSUS ACTUAL
07 78 CA          JZ       /IF THEY ARE EQUAL, JUMP BACK TO THE
07 79 70          QUEST1   /COMMAND MODE
07 7A 00          0
07 7B 21          LXIH     /BAD CHECKSUM, SO SAY SO ON THE
07 7C 69          BDCHKS   /PRINTER
07 7D 12          0
07 7E CD          CALL     /PRINT THE MESSAGE
07 7F 7A          TOUT
07 80 05          0
07 81 C3          JMP      /THEN ENTER THE COMMAND MODE
07 82 70          QUEST1
07 83 00          0

07 84 CD  NOTSUM, CALL
07 85 8C          ADD1     /TREAT THE CURRENT FRAME AND
07 86 07          0        /NEXT FRAME AS DATA
07 87 77          MOVMA    /THEN SAVE IT IN MEMORY
07 88 23          INXH     /INCREMENT THE STORAGE POINTER
07 89 C3          JMP      /AND GET ANOTHER FRAME
07 8A 69          BNXTIN
07 8B 07          0
07 8C CD  ADD1,   CALL     /GET 2 FRAMES, 1 BYTE
07 8D 9D          BBYTE1
07 8E 07          0
07 8F C3          JMP
07 90 95          ADDCHK+3
07 91 07          0
07 92 CD  ADDCHK, CALL
07 93 9A          BBYTE
```

```
07 94 07            0
07 95 F5            PUSHPSW /SAVE "A"
07 96 82            ADDD    /ADD THE CHECKSUM
07 97 57            MOVDA   /AND SAVE IT BACK IN "D"
07 98 F1            POPPSW  /GET THE DATA WORD BACK
07 99 C9            RET
07 9A CD  BBYTE,    CALL    /READ THE 2 MSB'S FROM THE TAPE
07 9B 0C            IOR
07 9C 18            0
07 9D 0F  BBYTE1,   RRC     /ROTATE THE BITS TO THE MSB'S
07 9E 0F            RRC
07 9F 4F            MOVCA   /SAVE THE TEMPORARY VALUE IN "C"
07 A0 CD            CALL
07 A1 0C            IOR     /THEN READ THE 6 LSB'S
07 A2 18            0
07 A3 81            ADDC    /ADD THE MSB'S
07 A4 C9            RET     /AND RETURN WITH THE VALUE IN "A"

          /THIS IS THE I/O TABLE THAT TEA WRITES INTO R/W
          /MEMORY.  TEA THEN ACCESSES THE I/O SUBROUTINES
          /BY CALLING THESE JUMP INSTRUCTIONS.

07 A5 C3  IOTAB,    JMP     /GET THE KEYBOARD'S FLAG
07 A6 51            INCHK
07 A7 06            0
07 A8 C3            JMP     /GET A CHARACTER BUT DO NOT
07 A9 6A            TTYI    /PRINT IT
07 AA 06            0
07 AB C3            JMP     /GET A CHARACTER AND PRINT IT
07 AC 76            TTYIN
07 AD 06            0
07 AE C3            JMP     /PUNCH THE CONTENT OF THE A REG.
07 AF 79            TTYOUT
07 B0 06            0
07 B1 C3            JMP     /READ A CHARACTER INTO THE A REG.
07 B2 56            IORDR
07 B3 06            0
07 B4 C3            JMP     /PRINT THE CONTENT OF THE A REG.
07 B5 79            TTYOUT
07 B6 06            0
```

```
            /THIS IS THE ASSEMBLER PORTION OF THE
            /TYCHON EDITOR-ASSEMBLER.

07 B7 CD  ASSEMB, CALL    /PRINT A CR AND LF.
07 B8 39          CRLF
07 B9 02          0
07 BA 3A          LDA     /GET THE HEX/OCTAL SWITCH
07 BB 3E          HORO
07 BC 18          0
07 BD FE          CPI     /IS IT IN THE HEX MODE?
07 BE 48          "H"
07 BF C2          JNZ     /NO, IT IS AN O FOR OCTAL
07 C0 C8          POS     /SO PRINT "OCTAL"
07 C1 07          0
07 C2 21          LXIH    /IT IS H, LOAD H&L WITH THE MEM-
07 C3 CE          HEXSRC  /ORY ADDRESS FOR THE "HEXADECIMAL"
07 C4 11          0       /MESSAGE
07 C5 C3          JMP     /NOW PRINT IT
07 C6 CB          HEXP
07 C7 07          0
07 C8 21  POS,    LXIH    /LOAD H&L WITH THE ADDRESS FOR THE
07 C9 DA          OCTSRC  /"OCTAL" MESSAGE.
07 CA 11          0
07 CB CD  HEXP,   CALL    /PRINT THE MESSAGE POINTED TO
07 CC 7A          TOUT    /BY REGISTER PAIR H
07 CD 05          0
07 CE 21  ASSMB1, LXIH    /NOW PRINT "M OR T?"
07 CF 4F          TAS
07 D0 11          0
07 D1 CD          CALL
07 D2 7A          TOUT
07 D3 05          0
07 D4 CD          CALL    /INPUT A TELETYPE CHARACTER
07 D5 06          IOECHO
07 D6 18          0
07 D7 32          STA     /SAVE THE CHARACTER IN "MORT"
07 D8 3D          MORT    /NO MATTER WHAT WAS TYPED IN.
07 D9 18          0
07 DA FE          CPI     /SEE IF IT WAS "M" FOR MEMORY
07 DB 4D          "M"
07 DC CA          JZ      /YES, IT WAS AN "M", CONTINUE
07 DD EC          MOK     /WITH THE ASSEMBLY PROCESS
07 DE 07          0
07 DF FE          CPI     /SEE IF IT WAS A "T" FOR TAPE
07 E0 54          "T"
07 E1 CA          JZ      /YES, IT WAS A "T", CONTINUE
07 E2 EC          MOK     /WITH THE ASSEMBLY PROCESS
07 E3 07          0
07 E4 FE          CPI     /SEE IF IT WAS A CTRL-C
07 E5 03          CTRLC   /IF IT WAS, GO TO THE "COMDEC" AT
07 E6 CA          JZ      /THE BEGINNING OF THE PROGRAM
07 E7 70          QUEST1  /AND SEE WHAT THE USER'S UP TO.
07 E8 00          0
07 E9 C3          JMP     /IF NOT T,M OR CNTRL-C, ASK "M OR T"
```

```
07 EA CE          ASSMBI  /AGAIN
07 EB 07          0
07 EC 21  MOK,    LXIH    /LOAD REGISTER PAIR H WITH THE STARTING
07 ED 5B          AUTO    /ADDRESS OF THE MESSAGE "AUTO-LOAD ?"
07 EE 11          0
07 EF CD          CALL    /PRINT THE MESSAGE
07 F0 7A          TOUT
07 F1 05          0
07 F2 CD          CALL    /GET THE USER'S RESPONSE
07 F3 06          IOECHO  /TO THE QUESTION.
07 F4 18          0
07 F5 32          STA     /SAVE THE RESPONSE FOR LATER.
07 F6 41          PORL
07 F7 18          0
07 F8 2A          LHLD    /LOAD H&L WITH THE STARTING ADDRESS
07 F9 22          TXTBUF  /OF THE TEXT BUFFER
07 FA 18          0
07 FB 3A          LDA     /ASSEMBLING FROM MEMORY OR TAPE?
07 FC 3D          MORT
07 FD 18          0
07 FE FE          CPI     /RESPONSE EQUAL TO ASCII T?
07 FF 54          "T"
08 00 C2          JNZ     /NOT FROM TAPE, SO THE ENTIRE PROGRAM
08 01 07          NOTAP   /MUST BE STORED IN MEMORY
08 02 08          0
08 03 24          INRH    /IF FROM TAPE, RESERVE 256 R-W LOCATIONS
08 04 C3          JMP     /FOR "ONE" LINE OF TEXT
08 05 0A          NOTAP1
08 06 08          0
08 07 2A  NOTAP,  LHLD    /GET THE HIGHEST ADDRESS USED BY
08 08 14          HIADD   /THE TEXT BUFFER.
08 09 18          0
08 0A 23  NOTAP1, INXH    /INCREMENT THE ADDRESS BY TWO
08 0B 23          INXH
08 0C 22          SHLD    /THIS IS WHERE THE SYMBOLIC ADDRESS
08 0D 39          SYMADD  /TABLE WILL BE STORED
08 0E 18          0
08 0F 22          SHLD    /SAVE THIS ADDRESS IN SYMADF (THE ADDRESS
08 10 3B          SYMADF  /OF THE END OF THE TABLE) ALSO.
08 11 18          0
08 12 36          MVIM    /SAVE A TERMINATOR AT THE BEGINNING
08 13 9B          BTERM   /OF THE SYMBOLIC ADDRESS TABLE.
08 14 2A          LHLD    /GET THE LAST ADDRESS OF R-W MEMORY
08 15 24          ENDRAM
08 16 18          0
08 17 22          SHLD    /SAVE THE ADDRESS AS THE BEGINNING
08 18 12          BYTAB   /OF THE DEFINED BYTE TABLE
08 19 18          0
08 1A 36          MVIM    /ALSO SAVE A 233 TERMINATOR
08 1B 9B          BTERM   /AT THE BEGINNING OF THE DB TABLE.
08 1C 3E  PASS1,  MVIA    /SAVE A 377 IN "PASS" TO INDICATE
08 1D FF          377     /THAT THE ASSEMBLER IS MAKING THE
08 1E 32          STA     /FIRST PASS THROUGH THE PROGRAM
08 1F 42          PASS    /BEING ASSEMBLED.
```

```
08 20 18            0
08 21 CD            CALL     /SET A NUMBER OF MEMORY ADDRESS
08 22 35            CLRREG   /POINTERS AND COUNTERS TO ZERO.
08 23 18            0
08 24 CD            CALL     /IF ASSEMBLING FROM TAPE, ASK "R ?"
08 25 4F            READY    /AND READ A SINGLE LINE OF TEXT.
08 26 0C            0        /IF FROM MEMORY, SIMPLY RETURN.
08 27 CD            CALL
08 28 AB            LOOK     /SET UP THE SYMBOLIC ADDRESS TABLE
08 29 0A            0
08 2A CD            CALL     /THEN PRINT THE NUMBER OF ERRORS
08 2B 9A            ERROUT   /THAT OCCURRED DURING THE FIRST
08 2C 0F            0        /PASS
08 2D CD            CALL     /PRINT A CARRIAGE RETURN AND LINE
08 2E 39            CRLF     /FEED AFTER IT.
08 2F 02            0
08 30 CD            CALL     /THEN CLEAR OUT SOME POINTERS AND
08 31 35            CLRREG   /COUNTERS SO THAT THE SECOND PASS
08 32 10            0        /CAN BE PERFORMED.
08 33 CD   PASS2,   CALL     /DURING PASS 2, WE PUNCH A TAPE.
08 34 4F            READY    /SEE IF THE USER'S READY.
08 35 0C            0
08 36 3E            MVIA     /P=PUNCH
08 37 50            "P"
08 38 32            STA      /SAVE A "P" IN PASS, TO INDICATE THAT
08 39 42            PASS     /THAT THE SECOND PASS IS BEING
08 3A 18            0        /PERFORMED.
08 3B CD            CALL     /PUNCH 10" OF PAPER TAPE LEADER
08 3C 55            LDR
08 3D 05            0
08 3E CD   MORMNE,  CALL     /NOW ASSEMBLE THE ENTIRE PROGRAM
08 3F 6D            MNEOK    /THIS IS PASS # 2
08 40 08            0
08 41 3E            MVIA     /TAPE HAS BEEN PUNCHED, PUNCH THE
08 42 C0            300      /CHECKSUM FLAG
08 43 CD            CALL     /AS 8 BITS PARALLEL ON
08 44 09            IOP      /THE PAPER TAPE).
08 45 18            0
08 46 3A            LDA      /NOW GET THE CHECKSUM
08 47 3F            CHKSUM
08 48 18            0
08 49 4F            MOVCA
08 4A CD            CALL     /AND PUNCH IT OUT
08 4B ED            BYTE2    /(BYTE2 AVOIDS THE ADD CHECKSUM SOFTWARE)
08 4C 0F            0
08 4D CD            CALL     /PUNCH ANOTHER 10" OF LEADER NOW
08 4E 55            LDR      /THAT THE ENTIRE PROGRAM HAS BEEN
08 4F 05            0        /ASSEMBLED.
08 50 CD            CALL     /TYPE OUT THE # OF ERRORS
08 51 9A            ERROUT   /THAT OCCURRED WHILE WE
08 52 0F            0        /WHERE PUNCHING OUT THE TAPE.
08 53 CD   PASS3,   CALL     /READY FOR THE THIRD PASS?
08 54 4F            READY
08 55 0C            0
```

```
06 56 CD          CALL     /YES, PRINT A CR AND LF.
06 57 39          CRLF
06 58 02          0
06 59 3E          MVIA
06 5A 4C          "L"      /L=LISTING OF THE ASSEMBLED PROGRAM
06 5B 32          STA
06 5C 42          PASS
06 5D 18          0
06 5E CD  DOIT,   CALL     /CLEAR OUT SOME POINTERS AND COUNTERS
06 5F 35          CLRREG
06 60 10          0
06 61 CD          CALL     /PRINT THE "HEADING INFORMATION" AND
06 62 3E          PAGE2    /THE PAGE NUMBER "01-001."
06 63 0F          0
06 64 CD          CALL     /THEN ASSEMBLE THE PROGRAM AGAIN.
06 65 6D          MNEOK
06 66 06          0
06 67 CD  SYML,   CALL     /TYPE OUT THE SYMBOLIC
06 68 EB          SYMOUT   /ADDRESS TABLE .
06 69 0D          0
06 6A C3          JMP      /AND GET ANOTHER COMMAND.
06 6B 70          QUEST1
06 6C 00          0

          /FIND THE MNEMONIC, CHECK TO SEE IF ITS VALID,
          /DETERMINE THE # OF BYTES AND CHECK THE FOR-
          /MAT OF THE OTHER BYTES, IF APPLICABLE.

06 6D 2A  MNEOK,  LHLD     /H&L POINT TO THE BEGINNING
06 6E 22          TXTBUF   /OF THE TEXT BUFFER.
06 6F 18          0
06 70 22          SHLD     /INITIALIZE TFCLIN TO THE SAME ADDRESS
06 71 2E          TFCLIN
06 72 18          0
06 73 AF  NM,     XRAA     /SET A AND B TO ZERO.
06 74 47          MOVBA
06 75 32          STA      /SET THE MNEMONIC VALUE TO 000, BEFORE
06 76 36          MNEVAL   /THE NEXT MNEMONIC IS INTERPRETED.
06 77 18          0
06 78 CD          CALL     /WE WILL EXIT FINDM UPON FINDING
06 79 54          FINDM    /THE END OF THE TEXT BUFFER OR
06 7A 06          0        /UPON FINDING A MNEMONIC.
06 7B FE          CPI      /IF A LINE CONTAINING ONLY A COMMENT
06 7C 9B          BTERM    /IS FOUND, WE WILL TYPE IT OUT IF AP-
06 7D C8          RZ       /PROPRIATE, AND GO ON TO THE NEXT LINE.
06 7E FE          CPI      /WAS THE DOLLAR SIGN TERMINATOR AT THE
06 7F 24          044      /END OF THE TEXT BUFFER FOUND?
06 80 C8          RZ       /YES, THEN RETURN
06 81 C3          JMP
06 82 F0          FMNOK    /WE FOUND A MNEMONIC, CHECK IT OUT !
06 83 06          0

          /"FINDM" IS USED TO FIND A MNEMONIC IN A LINE OF TEXT
```

```
08 84 3A  FINDM,  LDA     /WHERE IS THE SOURCE COMING FROM?
08 85 3D          MORT
08 86 18          0
08 87 FE          CPI     /IS IT COMING FROM TAPE?
08 88 54          "T"
08 89 C2          JNZ     /NO, ITS STORED IN MEMORY.
08 8A 92          FT
08 8B 08          0
08 8C 2A          LHLD    /ITS COMING FROM TAPE, SO ONLY ONE LINE
08 8D 22          TXTBUF  /WILL BE STORED IN MEMORY AT ONE TIME,
08 8E 18          0       /AND ALWAYS STARTING AT "TXTBUF"
08 8F C3          JMP
08 90 95          NXTCHR-1 /NOW PROCESS THE LINE OF TEXT
08 91 08          0

          /JUMP TO "FT" IF THE SOURCE IS IN MEMORY.

08 92 2A  FT,     LHLD    /H&L POINT TO THE FIRST CHARACTER IN
08 93 2E          TFCLIN  /A LINE (TEXT,FIRST CHARACTER (IN A)
08 94 18          0       /LINE).
08 95 2B          DCXH    /BACK THE ADDRESS UP BY ONE
08 96 23  NXTCHR, INXH    /GET A TEXT CHARACTER
08 97 7E  NXCHR1, MOVAM
08 98 FE          CPI
08 99 24          044     /DOLLAR SIGN?, (A TEXT BUFFER
08 9A C8          RZ      /TERMINATOR). YES, START ANOTHER PASS.
08 9B FE          CPI     /233 TERMINATOR (AT THE END OF THE
08 9C 9B          BTERM   /TEXT BUFFER)?, YES, START ANOTHER PASS.
08 9D C8          RZ
08 9E FE          CPI     /FOUND AN ASCII SPACE?
08 9F 20          040
08 A0 CA          JZ      /YES, THEN IGNORE IT AND GET
08 A1 96          NXTCHR  /ANOTHER CHARACTER FROM MEMORY.
08 A2 08          0
08 A3 FE          CPI
08 A4 2A          052     /* FOR AN ADDRESS ? (*=PSEUDO-OP)
08 A5 C2          JNZ     /(*IS SIMILAR TO "ORG") NOT AN *
08 A6 AE          NOSTAR
08 A7 08          0
08 A8 CD          CALL    /FOUND A *, SO PROCESS THE ADDRESS
08 A9 DC          ADDOK   /AND SAVE THE 16-BIT RESULT IN
08 AA 0C          0       /"ADDCNT."
08 AB C3          JMP     /THEN PROCESS THE NEXT LINE OF
08 AC 97          NXCHR1  /TEXT.
08 AD 08          0
```

```
08 AE FE  NOSTAR, CPI      /NOT AN *, IS IT A TAB?
08 AF 0A          012      /TAB=011 ON TAPE, 0-10 IN MEMORY
08 B0 DA          JC       /TAB ON TAPE OR IN MEMORY
08 B1 96          NXTCHR
08 B2 08          0        /SKIP THE TABS
08 B3 FE          CPI      /IS IT A SLASH FOR A COMMENT?
08 B4 2F          057
08 B5 CA          JZ       /YES, THEN IGNORE THE REMAINDER
08 B6 D5          ENDLN    /OF THE LINE OF TEXT.
08 B7 08          0
08 B8 FE  NOSLSH, CPI      /NOT A SLASH, IS IT A COMMA AT THE END
08 B9 2C          054      /OF A SYMBOLIC ADDRESS?
08 BA C2          JNZ      /NO, IS IT A CARRIAGE RETURN?
08 BB C2          NOCOMA
08 BC 08          0
08 BD 06          MVIB     /IT IS A COMMA, SO SET B=000.
08 BE 00          000
08 BF C3          JMP      /THEN EXAMINE THE NEXT CHARACTER IN
08 C0 96          NXTCHR   /THE LINE OF TEXT.
08 C1 08          0
08 C2 FE  NOCOMA, CPI      /NOT A COMMA, IS IT A CARRIAGE
08 C3 0D          CR       /RETURN ?
08 C4 CA          JZ       /YES, THEN ITS THE END OF THE LINE.
08 C5 D5          ENDLN
08 C6 08          0
08 C7 78          MOVAB    /MUST BE A LETTER IN A SYBOLIC ADDRESS
08 C8 FE          CPI      /OR A MNEMONIC TO GET HERE
08 C9 FF          377      /DOES B=377?
08 CA CA          JZ       /YES, THEN LOOK AT THE NEXT CHARACTER.
08 CB 96          NXTCHR
08 CC 08          0
08 CD 22          SHLD     /NO, THEN WE HAVE FOUND THE FIRST
08 CE 30          TFMLIN   /CHARACTER IN A MNEMONIC, SAVE ITS
08 CF 18          0        /ADDRESS
08 D0 06          MVIB     /SET THE B REGISTER TO 377
08 D1 FF          377
08 D2 C3          JMP      /AND EXAMINE THE REST OF THE CHARACTERS
08 D3 96          NXTCHR   /IN THE LINE OF TEXT.
08 D4 08          0
08 D5 78  ENDLN,  MOVAB    /IF NO MNEMONIC, B=000
08 D6 FE          CPI
08 D7 00          000
08 D8 C0          RNZ      /MUST HAVE FOUND A MNEMONIC, SO RETURN
08 D9 CD          CALL     /ITS A COMMENT.  SHOULD IT BE TYPED
08 DA E4          SPECL1   /OUT? (ONLY DURING PASS 3).
08 DB 08          0
08 DC 06  NOCOUT, MVIB     /CLEAR THE SENSE REGISTER
08 DD 00          000
08 DE CD          CALL     /AND FIND THE NEXT LINE OF TEXT (IF
08 DF A5          FINDQR   /ASSEMBLING FROM TAPE, THE 8080
08 E0 0D          0        /READS ANOTHER LINE).
08 E1 C3          JMP      /THEN PROCESS THIS LINE OF TEXT
08 E2 97          NXTCHR+1
08 E3 08          0
```

```
08 E4 3A  SPECL1, LDA     /FIND OUT WHAT THE PASS # IS.
08 E5 42          PASS
08 E6 18          0
08 E7 FE          CPI     /ITS PASS 3 IF "PASS" = "L"
08 E8 4C          "L"
08 E9 C0          RNZ     /NOT PASS 3, DON'T TYPE ANYTHING
08 EA CD          CALL    /PASS 3, SKIP OVER THE ADDRESS AND
08 EB 1B          SPCIT   /MNEMONIC VALUE COLUMNS
08 EC 10          0
08 ED C3          JMP     /NOW TYPE THE COMMENT. (THIS IS ALSO
08 EE 23          SPECL   /USED WHEN WE HAVE A PSEUDO-OP
08 EF 0C          0       /SUCH AS DB OR DW).

          /WE FOUND A MNEMONIC IN THE TEXT BUFFER, SO MAKE SURE
          /THAT IT IS A VALID MNEMONIC, BY CHECKING IT AGAINST
          /THE ENTRIES IN THE "PERMANENT SYMBOL TABLE."

08 F0 06  FMNOK,  MVIB    /B=# OF BYTES TO SKIP IN THE TABLE.
08 F1 02          002
08 F2 0E          MVIC    /C=NODE TERMINATION CHARACTER
08 F3 3D          075
08 F4 2A          LHLD    /LOAD H&L WITH THE ADDRESS OF THE
08 F5 30          TFMLIN  /FIRST CHARACTER IN THE MNEMONIC
08 F6 18          0       /STORED IN THE TEXT BUFFER.
08 F7 EB          XCHG
08 F8 21          LXIH
08 F9 77          MNELST  /SET H&L TO THE MNEMONIC TABLE.
08 FA 12          0
08 FB CD          CALL    /SEE IF THE MNEMONIC IS IN THE TABLE
08 FC 6E          SEARCH
08 FD 0B          0
08 FE 47          MOVBA   /SAVE THE 233 IF THE END OF THE TABLE
08 FF 3A          LDA
09 00 36          MNEVAL  /THE FOLLOWING WILL NOT BE "FOUND"
09 01 18          0       /IN THE TABLE: MOV'S,MVI'S,INRX,DCRX
09 02 FE          CPI     /IMMEDIATE MATH INSTRUCTIONS AND
09 03 40          100     /ALL REGULAR MATH INSTRUCTIONS.
09 04 CA          JZ
09 05 70          MOVE    /100 MEANS ITS A  MOV-TYPE INSTRU.
09 06 0A          0
09 07 FE          CPI
09 08 04          004     /004 OR 0X4 IS AN INCREMENT INSTRU.
09 09 CA          JZ      /NOW DETERMINE THE REGISTER INVOLVED
09 0A 65          MOVI
09 0B 0A          0
09 0C FE          CPI
09 0D 05          005     /005 OR 0X5 IS A DECREMENT INSTRUCTION
09 0E CA          JZ      /NOW DETERMINE THE REGISTER INVOLVED
09 0F 65          MOVI
09 10 0A          0
09 11 FE          CPI
09 12 06          006     /006 OR 0X6 IS A MOVE IMMEDIATE
09 13 CA          JZ      /DETERMINE THE REGISTER INVOLVED
```

```
09 14 65          MOVI
09 15 0A          0
09 16 FE          CPI
09 17 80          200      /IF THE VALUE IS BETWEEN 200 AND 271
09 18 DA          JC       /THEN ITS A MATH INSTRUCTION.
09 19 20          NOPART
09 1A 09          0
09 1B FE          CPI
09 1C B9          271
09 1D DA          JC       /OK, ITS EQUAL TO OR GREATER THAN 200
09 1E 85          MATH     /AND LESS THAN 271, SO ITS A MATH OR
09 1F 0A          0        /LOGICAL INSTRUCTION, GET THE REGISTER.
09 20 78  NOPART, MOVAB    /WAS THERE A TABLE TERMINATOR ?
09 21 FE          CPI
09 22 9B          BTERM
09 23 C2          JNZ      /NO, SO THE MNEMONIC IS STORED
09 24 3F          FOUND    /IN ITS ENTIRETY, IN THE TABLE
09 25 09          0
09 26 CD  NOMATH, CALL     /IF IT WASN'T IN THE MNEMONIC PER-
09 27 84          FINDM    /MANENT SYMBOL TABLE, WE GET HERE.
09 28 08          0
09 29 2A          LHLD     /GET THE ADDRESS OF THE FIRST CHARACTER
09 2A 30          TFMLIN   /OF THE "MNEMONIC" (IT MAY BE A SYM-
09 2B 18          0        /BOLIC ADDRESS).
09 2C EB          XCHG     /SEE IF ITS A SYMBOLIC ADDRESS
09 2D 2A          LHLD     /GET THE STARTING ADDRESS OF THE
09 2E 39          SYMADD   /SYMBOLIC ADDRESS TABLE THAT THE AS-
09 2F 18          0        /SEMBLER CREATED DURING PASS 1.
09 30 0E          MVIC     /LOAD C WITH THE STRING TERMINATOR
09 31 3D          075
09 32 06          MVIB     /LOAD B WITH THE NUMBER OF BYTES TO SKIP
09 33 03          003
09 34 CD          CALL     /SEARCH THE SYMBOLIC ADDRESS TABLE FOR
09 35 6E          SEARCH   /A MATCHING ENTRY.
09 36 0B          0
09 37 FE          CPI
09 38 9B          BTERM    /DID WE FIND THE END OF
09 39 C2          JNZ      /SYMBOLIC ADDRESS TABLE ?, NO
09 3A 18          NOSYMA   /SO WE MUST HAVE FOUND A
09 3B 0A          0        /SYMBOLIC ADDRESS
09 3C C3          JMP      /NOT A SYMBOLIC ADDRESS, SEE IF
09 3D D5          OB       /ITS A NUMBER OR A DEFINED BYTE
09 3E 09          0

09 3F 3A  FOUND,  LDA      /WE FOUND THE "MNEMMONIC" IN THE TABLE
09 40 36          MNEVAL   /FIRST SEE IF IS A "PSEUDO-OP", SUCH
09 41 18          0        /AS "DB" OR "DW."
09 42 FE          CPI
09 43 08          010      /DB OCTAL CODE
09 44 CA          JZ
09 45 94          DB       /YES, USER IS DEFINING A BYTE
09 46 10          0
09 47 FE          CPI
09 48 10          020      /DW OCTAL CODE
```

```
09 49 CA        JZ
09 4A 7F        DW      /YES, USER IS DEFINING A WORD
09 4B 10        0
09 4C 06        MVIB
09 4D FF        377
09 4E CD        CALL    /IT WASN'T A PSEUDO-OP, ITS A MNEMONIC.
09 4F A0        CRIO    /DO WE PUNCH THE VALUE ON TAPE
09 50 0D        0       /OR TYPE OUT THE LINE OF CODE ?
09 51 3A        LDA
09 52 36        MNEVAL  /FIND OUT HOW MANY BYTES THE INSTRUC-
09 53 18        0       /TION IS AND HOW MANY ADDITIONAL BYTES
09 54 47        MOVBA   /ARE REQUIRED BY THE INSTRUCTION
09 55 FE        CPI
09 56 C3        303     /UNCONDITIONAL JUMP
09 57 CA        JZ
09 58 BC        SYMOCT  /THE REMAINING BYTES CAN BE A SYMBOLIC
09 59 09        0       /ADDRESS OR TWO EIGHT-BIT NUMBERS.
09 5A FE        CPI
09 5B CD        315     /UNCONDITIONAL CALL
09 5C CA        JZ
09 5D BC        SYMOCT  /SAME AS A JUMP INSTRUCTION (ABOVE)
09 5E 09        0
09 5F FE        CPI
09 60 D3        323     /OUT INSTRUCTION
09 61 CA        JZ
09 62 D2        NMONLY  /A NUMBER OR DB CAN SPECIFY THE
09 63 09        0       /OUTPUT PORT.
09 64 FE        CPI
09 65 DB        333     /IN INSTRUCTION
09 66 CA        JZ
09 67 D2        NMONLY  /SAME AS THE OUTPUT INSTRUCTION (ABOVE).
09 68 09        0
09 69 E6        ANI
09 6A C7        307
09 6B FE        CPI
09 6C 01        001     /LXIB,D,H OR SP ?
09 6D CA        JZ      /MAYBE, PERFORM ANOTHER MASKING
09 6E AD        LXI     /OPERATION.
09 6F 09        0
09 70 FE        CPI
09 71 02        002     /DOUBLE PRECISION LOAD & STORES
09 72 CA        JZ      /MAYBE, PERFORM ANOTHER MASKING
09 73 B6        DPLS    /OPERATION.
09 74 09        0
09 75 FE        CPI
09 76 C2        302     /CONDITIONAL JUMPS
09 77 CA        JZ
09 78 BC        SYMOCT  /SYMBOLIC ADDRESS OR NUMBERS
09 79 09        0
09 7A FE        CPI
09 7B C4        304     /CONDITIONAL CALLS
09 7C CA        JZ
09 7D BC        SYMOCT  /SYMBOLIC ADDRESS OR NUMBERS.
09 7E 09        0
```

```
09 7F FE          CPI
09 80 06          006      /MOVE IMMEDIATE ?
09 81 CA          JZ
09 82 8F          OCTQ     /NUMBER, QUOTED CHARACTER OR DB.
09 83 09          0
09 84 FE          CPI
09 85 C6          306      /IMMEDIATE MATH
09 86 CA          JZ
09 87 8F          OCTQ     /NUMBER, QUOTED CHARACTRE OR DB.
09 88 09          0
09 89 2A  GENENT, LHLD     /ITS SIMPLY A SINGLE BYTE INSTRUCTION,
09 8A 2E          TFCLIN   /SO GET THE ADDRESS WHERE THE NEXT
09 8B 18          0        /LINE OF TEXT IS STORED
09 8C C3          JMP      /AND CONTINUE THE ASSEMBLY PROCESS
09 8D 73          NM
09 8E 08          0
```

```
        /"OCTQ" IS USED ONLY WHEN AN IMMEDIATE
        /INSTRUCTION IS ASSEMBLED.  WE WANT TO SEE IF
        /THE SECOND (DATA) BYTE IS A NUMBER, QUOTED
        /CHARACTERS, SUCH AS "2" OR "?", OR A DEFINED BYTE.

09 8F CD  OCTQ,   CALL
09 90 84          FINDM    /FIND MNEMONIC OR DATA VALUE
09 91 08          0
09 92 2A          LHLD     /H&L POINT TO THE FIRST CHARACTER
09 93 30          TFMLIN
09 94 18          0
09 95 7E          MOVAM    /GET THE FIRST CHARACTER
09 96 FE          CPI
09 97 22          042      /IS IT A QUOTE ?
09 98 C2          JNZ      /NO, THEN IS THE DATA BYTE A NUMBER ?
09 99 D5          OB
09 9A 09          0
09 9B 23  QUOTE,  INXH     /WE FOUND ONE ", IS THERE ANOTHER "?
09 9C 46          MOVBM    /SAVE THE CHARACTER JUST AFTER THE "
09 9D 23          INXH
09 9E 7E          MOVAM    /THIS SHOULD BE THE 2ND "
09 9F FE          CPI
09 A0 22          042      /IS THIS THE 2ND " ?
09 A1 CA          JZ       /YES, THEN B CONTAINS THE ASCII CHARACTER
09 A2 A9          OKQ2
09 A3 09          0
09 A4 CD          CALL     /THE 2ND QUOTE WAS MISSING, ITS AN
09 A5 2C          UDOUT    /UNDEFINED SYMBOL (UD)
09 A6 10          0        /SO TYPE THE "UD" ERROR MESSAGE"
09 A7 06          MVIB     /DATA FOR THE IMMEDIATE
09 A8 00          000      /INSTRUCTION IS SET TO 000
09 A9 78  OKQ2,   MOVAB    /A NOW ='S THE QUOTED CHARACTER
09 AA C3          JMP
09 AB 48          ZEOUT1
09 AC 0A          0

09 AD 78  LXI,    MOVAB    /ALL LXI INSTRUCTION HAVE AN EVEN
09 AE E6          ANI      /MIDDLE BYTE (0,2,4 OR 6)
09 AF 08          010
09 B0 C2          JNZ      /IF D4=1, ITS NOT AN LXI INSTRUCTION
09 B1 89          GENENT   /SO TREAT IT AS A SINGLE BYTE INSTRUCTION
09 B2 09          0
09 B3 C3          JMP      /OK, ITS A LXIB,D,H OR SP
09 B4 BC          SYMOCT   /SO SEE IF A SYMBOLIC ADDRESS
09 B5 09          0        /FOLLOWS OR OCTAL OR HEX NUMBERS

09 B6 78  DPLS,   MOVAB    /ALL THE DOUBLE PRECISION LOADS HAVE
09 B7 FE          CPI      /MNEMONIC VALUES GREATER THAN 041
09 B8 22          042      /SUCH AS 042,052,062 AND 072 (OCTAL)
09 B9 DA          JC       /NOT AN SHLD, LHLD, STA OR LDA
09 BA 89          GENENT
09 BB 09          0
09 BC CD  SYMOCT, CALL     /FIND THE OCTAL # OR SYMBOLIC ADDRESS
09 BD 84          FINDM    /IN THE SECOND BYTE
```

```
09 BE 08          0
09 BF 2A          LHLD
09 C0 30          TFMLIN
09 C1 18          0
09 C2 2B          DCXH
09 C3 CD          CALL     /SEE IF NUMBERS ARE
09 C4 27          THRNMB   /AFTER THE INSTRUCTION
09 C5 0D          0
09 C6 CA          JZ       /NO, NO VALID NUMBERS FOLLOW
09 C7 FB          CHKSYM   /SEE IF ITS A SYMBOLIC ADDRESS
09 C8 09          0
09 C9 79  LOWOK,  MOVAC    /THERE WERE #'S, C=THE LOW ADDRESS
09 CA 32          STA
09 CB 36          MNEVAL
09 CC 18          0
09 CD 06          MVIB     /PUNCH OUT OR PRINT THE LO ADDRESS BYTE
09 CE FF          377
09 CF CD          CALL
09 D0 A0          CRIO     /FIND THE NEXT LINE CONTAINING THE HI ADD.
09 D1 0D          0
09 D2 CD  NMONLY, CALL     /FIND THE "MNEMONIC" IN THE LINE
09 D3 84          FINDM
09 D4 08          0
09 D5 2A  OB,     LHLD
09 D6 30          TFMLIN
09 D7 18          0
09 D8 EB          XCHG     /NOW SEE IF THE BYTE IS DEFINED
09 D9 2A          LHLD     /IN THE "BYTAB"
09 DA 12          BYTAB
09 DB 18          0
09 DC 06          MVIB     /THIS IS THE "SKIP COUNTER"
09 DD 02          002
09 DE 0E          MVIC     /C=TERMINATOR=EQUAL SIGN
09 DF 3D          075
09 E0 CD          CALL
09 E1 6E          SEARCH   /SEE IF ITS IN THE TABLE
09 E2 0B          0
09 E3 FE          CPI      /FIND THE END OF THE TABLE ?
09 E4 9B          BTERM    /IF NOT, THERE IS A SIMILAR ENTRY
09 E5 C2          JNZ
09 E6 4B          NZ       /IT WAS IN THE TABLE, VALUE
09 E7 0A          0        /WAS STORED IN "MNEVAL" BY "SEARCH"
09 E8 2A          LHLD     /IT WASN'T IN THE TABLE
09 E9 30          TFMLIN   /SEE IF ITS AN OCTAL OR HEX NUMBER
09 EA 18          0
09 EB 2B          DCXH     /IT MUST BE A NUMBER ALSO
09 EC CD          CALL
09 ED 27          THRNMB   /INTERPRETE THE NUMBER
09 EE 0D          0
09 EF C2          JNZ  .   /IT WAS VALID, THE VALUE IS IN "C"
09 F0 F7          OCTEND
09 F1 09          0
09 F2 CD          CALL     /IT WAS NOT A VALID NUMBER
09 F3 0E          BNOUT    /TYPE THE "BAD NUMBER " (BN)
```

```
09 F4 0D            0        /ERROR MESSAGE, AND SET C=000
09 F5 0E            MVIC     /C IS THE THRNMB REGISTER
09 F6 00            000
09 F7 79  OCTEND,   MOVAC
09 F8 C3            JMP
09 F9 48            ZEOUT1   /PUNCH OR PRINT THE NUMERIC VALUE
09 FA 0A            0

09 FB 2A  CHKSYM,   LHLD     /SEE IF THE WORD IS
09 FC 30            TFMLIN   /A SYMBOLIC ADDRESS
09 FD 18            0
09 FE EB            XCHG     /D&E POINT TO THE CHARACTER IN THE BUFFER
09 FF 2A            LHLD
0A 00 39            SYMADD
0A 01 18            0
0A 02 0E            MVIC     /THIS IS THE STRING TERMINATOR
0A 03 3D            075
0A 04 06            MVIB     /THIS IS THE NUMBER OF BYTES TO SKIP
0A 05 03            003
0A 06 CD            CALL
0A 07 6E            SEARCH   /SEARCH THE SYMBOLIC ADDRESS TABLE
0A 08 0B            0        /FOR A MATCH WITH THE TEXT BUFFER WORD
0A 09 FE            CPI
0A 0A 9B            BTERM    /END OF THE BUFFER YET?
0A 0B C2            JNZ
0A 0C 18            NOSYMA   /IT WAS IN THE TABLE, THE
0A 0D 0A            0        /LO ADD. IS IN MNEVAL, HI IN MNEVAL+1
0A 0E 2E  BADHL,    MVIL     /THERE WAS NO SYMBOL IN THE
0A 0F 00            000      /TABLE, SO SET THE ADDRESS
0A 10 26            MVIH     /TO 000 000
0A 11 00            000
0A 12 22            SHLD     /AND SAVE IT FOR USE LATER.
0A 13 36            MNEVAL
0A 14 18            0
0A 15 CD            CALL     /PRINT A "UD AT LINE #"
0A 16 2C            UDOUT
0A 17 10            0
0A 18 06  NOSYMA,   MVIB     /SET B TO 377, SO BOTH THE ADDRESS
0A 19 FF            377      /AND LINE COUNT ARE INRC. BY "CRIO"
0A 1A CD            CALL     /PUT OUT THE LOW ADDRESS ON TAPE
0A 1B A0            CRIO     /OR LISTING WHICHEVER
0A 1C 0D            0
0A 1D CD  ZNEXT,    CALL     /WE FOUND THE SYMBOLIC ADDRESS ON
0A 1E 84            FINDM    /THE 2ND LINE, MAKE SURE THERE IS A 0
0A 1F 08            0        /ON THE 3RD LINE OF TEXT
0A 20 FE            CPI
0A 21 9B            BTERM
0A 22 CA            JZ       /END OF TEXT BUFFER AND NO 0 ?
0A 23 3A            NZER     /YES, THEN PRINT A ERROR MESSAGE
0A 24 0A            0        /(NZ AT LINE #) AND KEEP GOING.
0A 25 FE            CPI
0A 26 24            044      /DOLLAR SIGN
0A 27 CA            JZ       /END OF TEXT BUFFER AND NO 0 ?
0A 28 3A            NZER     /YES, THEN PRINT AN ERROR MESSAGE
```

```
0A 29 0A            0        /(NZ AT LINE #) AND KEEP GOING.
0A 2A 2A            LHLD     /OK, SEE IF THERE REALLY WAS A 0
0A 2B 30            TFMLIN
0A 2C 18            0
0A 2D 2B            DCXH
0A 2E CD            CALL     /CONVERT THE ASCII "0" TO BINARY
0A 2F 27            THRNMB
0A 30 0D            0
0A 31 CA            JZ       /THERE WAS NO ZERO AFTER THE SYMBOLIC
0A 32 3A            NZER     /SO PRINT THE ERROR MESSAGE "NZ AT
0A 33 0A            0        /#" AND THEN CONTINUE ON
0A 34 79            MOVAC    /GET THE CONVERSION VALUE
0A 35 FE            CPI      /IS IT EQUAL TO ZERO?
0A 36 00            000
0A 37 CA            JZ       /YES, THEN EVERYTHING IS OK.
0A 38 45            ZEOUT
0A 39 0A            0
0A 3A E5  NZER,     PUSHH    /THERE WAS NO 0, SO TYPE THE "NZ"
0A 3B 21            LXIH     /ERROR MESSAGE.
0A 3C 6C            NZERO
0A 3D 11            0
0A 3E CD            CALL
0A 3F 0E            LISTI
0A 40 0F            0
0A 41 E1            POPH
0A 42 C3            JMP      /NO 0, SO WE HAVE H&L POINTING TO
0A 43 89            GENENT   /TO THE NEXT LINE OF CODE !
0A 44 09            0
0A 45 3A  ZEOUT,    LDA      /THERE WAS A 0, SO TRANSFER THE HI
0A 46 37            MNEVAL+1 /ADDRESS TO "MNEVAL".  THE CONTENTS OF
0A 47 18            0        /"MNEVAL" IS WHAT GETS PUNCHED OUT OR
0A 48 32  ZEOUT1,   STA      /LISTED, THATS WHY WE HAVE TO DO
0A 49 36            MNEVAL   /THIS DATA TRANSFER
0A 4A 18            0
0A 4B 06  NZ,       MVIB     /LOAD B WITH 377 SO THAT CRIO IN-
0A 4C FF            377      /CREMENTS THE ADDRESS AND LINE CNT.
0A 4D CD            CALL     /PUNCH OR LIST THE VALUE THEN
0A 4E A0            CRIO     /GET THE NEXT LINE OF TEXT
0A 4F 0D            0
0A 50 C3            JMP      /THEN CONTINUE THE ASSEMBLY PROCESS
0A 51 8°            GENENT
0A 52 09            0

0A 53 2A  DEST,     LHLD     /H&L POINT TO THE ASCII CHARACTER
0A 54 43            REG      /THAT REPRESENTS THE DESTINATION REGISTER.
0A 55 18            0
0A 56 CD            CALL
0A 57 8B            VALLET   /IF NOT VALID, TYPE A "UD"
0A 58 0A            0        /ERROR MESSAGE
0A 59 07            RLC .    /GET THE DESTINATION REGISTER VALUE
0A 5A 07            RLC      /AND ROTATE IT TO THE THREE
0A 5B 07            RLC      /MIDDLE BITS.
0A 5C 47            MOVBA
0A 5D 3A            LDA      /THEN GET THE MNEMONIC'S "VALUE"
```

```
0A 5E 36          MNEVAL
0A 5F 18          0
0A 60 80          ADDB     /ADD THE REGISTER VALUE TO IT.
0A 61 32          STA      /AND SAVE THE RESULT FOR USE
0A 62 36          MNEVAL   /BY OTHER SECTIONS OF THE ASSEMBLER.
0A 63 18          0
0A 64 C9          RET
```

```
0A 65 CD  MOVI,   CALL
0A 66 53          DEST     /GET THE REGISTER DESTINATION CODE
0A 67 0A          0
0A 68 CD          CALL
0A 69 59          VALTRM   /IS THERE A VALID TERMINATOR ?
0A 6A 0B          0        /(A CR,TAB, SPACE OR SLASH)
0A 6B 06          MVIB
0A 6C FF          377      /SET THE SENSE REGISTER TO 377
0A 6D C3          JMP      /(MNEMONIC FOUND)
0A 6E 3F          FOUND
0A 6F 09          0

0A 70 CD  MOVE,   CALL     /THERE IS A DESTINATION AND SOURCE
0A 71 53          DEST     /REGISTER FOR THE MOVE INSTRUCTIONS
0A 72 0A          0        /FIRST GET THE DESTINATION CODE
0A 73 23          INXH
0A 74 CD  MATH1,  CALL     /THEN GET THE SOURCE (THE MATH
0A 75 8B          VALLET   /INSTRUCTIONS ENTER HERE, 1 REGISTER).
0A 76 0A          0
0A 77 47          MOVBA    /SAVE THE REGISTER CODE IN "B"
0A 78 3A          LDA      /GET THE MNEMONIC CODE
0A 79 36          MNEVAL
0A 7A 18          0
0A 7B 80          ADDB     /ADD THE REGISTER CODE
0A 7C 32          STA      /AND SAVE THE NEW MNEMONIC CODE
0A 7D 37          MNEVAL+1
0A 7E 18          0
0A 7F CD          CALL     /SEE IF THERE WAS A CR, SPACE,
0A 80 59          VALTRM   /TAB OR SLASH AFTER THE MNEMONIC
0A 81 0B          0
0A 82 C3          JMP
0A 83 45          ZEOUT
0A 84 0A          0

0A 85 2A  MATH,   LHLD     /IS A MATH INSTRUCTION (ADD, ADC, SUB, SBB,
0A 86 43          REG      /CMP, ORA, ANA OR CMP), SO GET THE
0A 87 18          0        /ADDRESS THAT POINTS TO THE REGISTER.
0A 88 C3          JMP      /AND DETERMINE THE CODE FOR IT.
0A 89 74          MATH1
0A 8A 0A          0

0A 8B 46  VALLET, MOVEM    /GET THE ASCII CHARACTER FOR THE REGISTER
0A 8C E5          PUSHH    /SAVE H&L
0A 8D 21          LXIH     /LOAD H&L WITH THE BASE ADDRESS
0A 8E BD          ALLET    /OF THE TABLE CONTAINING THE
0A 8F 11          0        /VALID ASCII REGISTER CODES.
0A 90 11          LXID     /LOAD REGISTER PAIR D WITH THE BASE
0A 91 C6          ALLETV   /ADDRESS OF THE TABLE THAT CONTAINS
0A 92 11          0        /THE VALUES (0-7) FOR THE REGISTERS.
0A 93 7E  TCBCHR, MOVAM    /GET AN ASCII REG. VALUE FROM THE TABLE
0A 94 FE          CPI      /END OF THE TABLE?
0A 95 FF          377
0A 96 CA          JZ       /YES, THEN THE REGISTER SPECIFIED IN
0A 97 A5          NOREG    /THE MNEMONIC IS NOT A VALID
```

```
0A 98 0A            0        /REGISTER
0A 99 B8            CMPB     /TABLE VALUE= MNEMONIC VALUE?
0A 9A CA            JZ       /YES, THEN GET THE VALUE FOR
0A 9B A2            LETOK    /THE REGISTER (0-7).
0A 9C 0A            0
0A 9D 23            INXH     /NO, INCREMENT THE REGISTER ADDRESS
0A 9E 13            INXD     /INCREMENT THE REGISTER VALUE ADDRESS
0A 9F C3            JMP      /AND TRY TO MAKE A MATCH
0A A0 93            TCBCHR
0A A1 0A            0
0A A2 1A  LETOK,    LDAXD    /GET THE REGISTER'S VALUE
0A A3 E1            POPH     /POP THE TEXT BUFFER ADDRESS
0A A4 C9            RET      /AND RETURN
0A A5 E1  NOREG,    POPH     /POP THE TEXT BUFFER POINTER
0A A6 CD            CALL     /PRINT "UD AT LINE #" BECAUSE
0A A7 2C            UDOUT    /AN INVALID REGISTER WAS
0A A8 10            0        /SPECIFIED IN THE MNEMONIC
0A A9 AF            XRAA     /SET THE "REGISTER VALUE" TO ZERO
0A AA C9            RET      /AND RETURN.

          /LOOK SETS UP THE SYMBOLIC ADDRESS TABLE.
          /IT FINDS ALL OF THE USER DEFINED SYMBOLIC
          /ADDRESSES AND SAVES THEM IN THE "SYMADD"
          /TABLE ALONG WITH AN EQUAL SIGN AND THE 2 BYTE
          /(16 BIT) HI AND LO ADDRESS. A 233 IS SAVED AT
          /THE END OF THE TABLE.

0A AB 2A  LOOK,     LHLD     /H&L POINT TO THE BEGINNING
0A AC 22            TXTBUF   /OF THE TEXT BUFFER
0A AD 18            0
0A AE 22            SHLD
0A AF 2E            TFCLIN   /SAVE THE POINTER TO THE TEXT,
0A B0 18            0        /FIRST CHARACTER (IN THE) LINE (TFCLIN)
0A B1 06            MVIB     /B IS USED TO SENSE WHETHER A SYMBOLIC
0A B2 00            000      /ADDRESS WAS DEFINED IN A LINE OF TEXT.
0A B3 2B            DCXH
0A B4 23  LOOK1,    INXH
0A B5 7E            MOVAM
0A B6 FE            CPI
0A B7 24            044      /044 IS AN ASCII DOLLAR SIGN.  IT
0A B8 C8            RZ       /INDICATES THE END OF THE TEXT BUFFER.
0A B9 FE            CPI
0A BA 9B            BTERM    /IF ASSEMBLING FROM MEMORY, A 233
0A BB C8            RZ       /IS AT THE END OF THE TEXT BUFFER
0A BC FE            CPI      /THE 012 IS FOR LOOKING FOR TABS
0A BD 0A            012      /011 ON TAPE, 0-10 IN MEMORY
0A BE DA            JC
0A BF B4            LOOK1
0A C0 0A            0
0A C1 FE            CPI      /FOUND A SLASH (FOR A COMMENT) IN THE
0A C2 2F            057      /TEXT BUFFER ?
0A C3 C2            JNZ      /NO, SEE IF ITS A ASTERISK.
0A C4 CC            NOSS
0A C5 0A            0
```

```
0A C6 CD  CROK,   CALL    /FOUND A COMMENT, SO FIND THE NEXT LINE
0A C7 A5          FINDCR  /OF TEXT
0A C8 0D          0
0A C9 C3          JMP     /AND TEST IT FOR A SYMBOLIC ADDRESS
0A CA B5          LOOK1+1
0A CB 0A          0
0A CC FE  NOSS,   CPI     /NOT A COMMENT, IS IT A
0A CD 2A          052     /DEFINED ADDRESS (*)?
0A CE C2          JNZ     /NO, NOT AN ADDRESS, SEE IF IT IS
0A CF D7          NOSR    /A SPACE.
0A D0 0A          0
0A D1 CD          CALL    /ITS AN *, SO PROCESS THE ADDRESS,
0A D2 DC          ADDOK   /SAVE THE 16-BIT RESULT IN "ADDCNT"
0A D3 0C          0       /AND PROCESS THE NEXT LINE OF TEXT
0A D4 C3          JMP
0A D5 B5          LOOK1+1
0A D6 0A          0

0A D7 FE  NOSR,   CPI     /NOT AN *, IS IT A SPACE?
0A D8 20          040
0A D9 CA          JZ      /YES, THEN IGNORE IT AND GET THE NEXT
0A DA B4          LOOK1   /CHARACTER FROM THE SAME LINE OF TEXT
0A DB 0A          0
0A DC FE          CPI     /NOT A SPACE, IS IT A CARRIAGE RETURN?
0A DD 0D          CR
0A DE CA          JZ      /YES, THEN FIND THE NEXT LINE OF
0A DF C6          CROK    /TEXT AND PROCESS IT.
0A E0 0A          0
0A E1 FE  NOCR,   CPI     /NOT A CARRIAGE RETURN, IS IT A COMMA?
0A E2 2C          054
0A E3 CA          JZ      /YES, THEN PROCESS THE SYMBOL BELOW.
0A E4 EB          SYMCMA
0A E5 0A          0
0A E6 06          MVIB    /NONE OF THE ABOVE, SO SET THE
0A E7 FF          377     /B REGISTER TO 377.
0A E8 C3          JMP     /AND CONTINUE EXAMINING THIS
0A E9 B4          LOOK1   /SAME LINE OF TEXT.
0A EA 0A          0

          /"SYMCMA" CHECKS THE CONTENT OF THE SYMBOLIC ADDRESS
          /TABLE TO SEE IF THE SAME SYMBOLIC ADDRESS
          /HAS ALREADY BEEN STORED IN THE TABLE (THIS IS AN ERROR
          /CONDITION; REDEFINITION; THE SAME SYMBOLIC ADDRESS
          /HAS BEEN USED AT LEAST TWICE).  IF THE SYMBOLIC ADDRESS
          /IS NOT IN THE TABLE ALREADY, THE 8080 ADDS IT TO THF
          /TABLE, ALONG WITH THE ASCII VALUE FOR AN =, AND
          /THE 16-BIT CONTENT OF "ADDCNT."

0A EB 2A  SYMCMA, LHLD    /LOAD REGISTER PAIR H WITH THE ADDRESS
0A EC 2E          TFCLIN  /FOR THE FIRST CHARACTER IN THE SYM-
0A ED 18          0       /BOLIC ADDRESS.
0A EE EB          XCHG
0A EF 06          MVIB    /NUMBER OF BYTES TO SKIP
0A F0 03          003
```

```
0A F1 CD          CALL
0A F2 32          CHKTB1  /CHECK THE SYMBOLIC ADDRESS TABLE
0A F3 11          0       /FOR THE SAME ENTRY. IF SO, ERROR
0A F4 CA          JZ      /IT WASN'T IN THE TABLE, SO
0A F5 FF          NOSYM   /ENTER IT.
0A F6 0A          0
0A F7 06  SYMIN,  MVIB    /NOW GET THE NEXT
0A F8 FF          377     /LINE OF TEXT.
0A F9 CD  FNL,    CALL    /INCREMENT THE ADDRESS AND LINE
0A FA A5          FINDCR  /COUNTER
0A FB 0D          0
0A FC C3          JMP     /AND CHECK THE NEXT LINE OF TEXT
0A FD B5          LOOK1+1 /FOR ANY DEFINED SYMBOLIC ADDRESSES
0A FE 0A          0

0A FF 2A  NOSYM,  LHLD    /THE SYMBOLIC ADDRESS WASN'T IN THE
0B 00 2E          TFCLIN  /TABLE, SO WE WILL ENTER IT IN TO
0B 01 18          0       /THE TABLE RIGHT NOW.
0B 02 EB          XCHG
0B 03 2A          LHLD    /GET THE LAST ADDRESS USED FOR STORING
0B 04 3B          SYMADF  /A SYMBOLIC ADDRESS (INITIALLY EQUAL TO
0B 05 18          0       /SYMADD).
0B 06 1A  SAVMOR, LDAXD   /D&E POINT TO THE TEXT BUFFER
0B 07 FE          CPI     /FIND THE COMMA AFTER THE SYMBOLIC
0B 08 2C          054     /ADDRESS IN THE TEXT BUFFER YET ?
0B 09 CA          JZ      /YES, THEN ENTER AN = SIGN AND
0B 0A 2F          ESA     /THE LO AND HI ADDRESS WHERE IT
0B 0B 0B          0       /WAS DEFINED.
0B 0C FE          CPI     /FIND THE SPACE REQUIRED IN THE
0B 0D 20          040     /FORMAT FOR THE "DW" PSEUDO-OP ?
0B 0E CA          JZ      /YES, THEN CALCULATE THE ADDRESS
0B 0F 4A          SPECDF  /AND ENTER IT AND AN = SIGN IN
0B 10 0B          0       /THE SYMBOLIC ADDRESS TABLE
0B 11 77          MOVMA   /SAVE THE TEXT CHARACTER IN THE SYM. TABLE
0B 12 23          INXH    /INCREMENT THE TEXT AND TABLE POINTERS
0B 13 13          INXD
0B 14 D5          PUSHD
0B 15 EB          XCHG
0B 16 2A          LHLD    /HAS THE SYMBOLIC ADDRESS TABLE
0B 17 12          BYTAB   /RUN INTO THE "DB" TABLE ?
0B 18 18          0
0B 19 7D          MOVAL
0B 1A 93          SUBE
0B 1B 47          MOVBA
0B 1C 7C          MOVAH
0B 1D 9A          SBBD
0B 1E EB          XCHG
0B 1F D1          POPD
0B 20 D2          JNC     /NOT YET, ENTER ANOTHER TEXT CHARACTER
0B 21 06          SAVMOR  /INTO THE SYMBOLIC ADDRESS TABLE
0B 22 0B          0
0B 23 21  MM,     LXIH    /YES, THE TWO TABLES HAVE RUN INTO
0B 24 72          ME      /EACH OTHER, SO WE NEED MORE MEMORY.
0B 25 11          0       /THIS IS A FATAL ERROR, THE ASSEMBLY
```

```
0B 26 CD  FATAL,  CALL    /PROCESS CAN NOT CONTINUE IF THIS
0B 27 0E          LISTI   /OCCURS.
0B 28 0F          0
0B 29 31          LXISP   /RESET THE STACK POINTER
0B 2A A0          STACK
0B 2B 18          0
0B 2C C3          JMP     /THE "ME" ERROR IS FATAL TO THE
0B 2D 6B          QUEST   /ASSEMBLY PROCESS, GOTO THE COMMAND
0B 2E 00          0       /DECODER.

0B 2F 36  ESA,    MVIM    /SAVED ALL OF THE CHARACTERS,
0B 30 3D          075     /SAVE AN = SIGN TERMINATOR
0B 31 23          INXH    /NOW GET THE ADDRESS AND SAVE IT ALSO.
0B 32 EB          XCHG
0B 33 2A          LHLD
0B 34 34          ADDCNT  /GET THE "CURRENT ADDRESS" AND ASSIGN IT
0B 35 18          0       /TO THE SYMBOLIC ADDRESS WE JUST ENTERED.
0B 36 CD          CALL    /NOW SAVE THE VALUE IN THE TABLE
0B 37 3C          SAVADD
0B 38 0B          0
0B 39 C3          JMP     /NOW FIND THE NEXT LINE OF CODE
0B 3A F7          SYMIN   /AND ASSEMBLE IT.
0B 3B 0A          0

0B 3C 7D  SAVADD, MOVAL
0B 3D 12          STAXD   /GET THE CURRENT ADDRESS AND SAVE IT
0B 3E 13          INXD
0B 3F 7C          MOVAH
0B 40 12          STAXD
0B 41 13          INXD
0B 42 3E          MVIA    /NOW SAVE A 233 TERMINATOR AFTER
0B 43 9B          BTERM   /THE ADDRESS TO INDICATE THE
0B 44 12          STAXD   /END OF THE TABLE
0B 45 EB          XCHG
0B 46 22          SHLD    /SAVE THE LAST ADDRESS USED. THIS WILL
0B 47 3B          SYMADF  /BE WHERE THE NEXT ENTRY STARTS.
0B 48 18          0
0B 49 C9          RET

0B 4A 36  SPECDF, MVIM    /SAVE A = SIGN ALSO
0B 4B 3D          075
0B 4C 23          INXH    /INCREMENT PAST THE = SIGN
0B 4D EB          XCHG    /H&L ARE TEXT, D&E ARE TABLE ADDRESSES
0B 4E D5          PUSHD
0B 4F CD          CALL    /NOW CALCULATE THE ADDRESS THAT
0B 50 C4          ADBYTE  /THE USER ASSIGNED TO THE SYMBOL
0B 51 0C          0
0B 52 D1          POPD    /GET THE TABLE ADDRESS BACK
0B 53 CD          CALL    /SAVE IT IN THE TABLE
0B 54 3C          SAVADD
0B 55 0F          0
0B 56 C3          JMP     /AND THEN RE-ENTER THE "PSEUDO-OP"
0B 57 23          PSEUDO  /PROCESSING ROUTINE
0B 58 11          0
```

```
          /VALTRM IS USED TO SEE IF THERE ARE ANY LEFTOVERS
          /AT THE "END" OF THE PARTIAL MNEMONICS, OCTAL #'S ETC.
          /THIS PREVENTS MOVABC,123E,INRBC AND SO ON.
          /BY CHECKING FOR A VALID TERMINATOR

0B 59 23  VALTRM, INXH     /INCREMENT PAST THE REGISTER IN MEM.
0B 5A 7E          MOVAM    /GET THIS CHARACTER
0B 5B FE          CPI      /A CR AFTER THE REGISTER?
0B 5C 0D          CR       /A CR IS A VALID TERMINATOR
0B 5D C8          RZ       /YES, THEN ITS TERMINATED CORRECTLY.
0B 5E FE          CPI      /SLASH FOR A COMMENT?
0B 5F 2F          057
0B 60 C8          RZ       /YES, THEN RETURN
0B 61 FE          CPI
0B 62 20          040      /FOUND A SPACE ?
0B 63 CA          JZ       /YES TRY THE NEXT CHARACTER IN MEMORY
0B 64 59          VALTRM
0B 65 0B          0
0B 66 FE          CPI
0B 67 0A          012      /THE SAME GOES FOR TABS SINCE WE
0B 68 DA          JC       /CAN USE MULTIPLE TABS
0B 69 59          VALTRM
0B 6A 0B          0
0B 6B C3          JMP      /THERE IS AN INVALID TERMINATOR AFTER
0B 6C 2C          UDOUT    /THE LAST REGISTER, SO PRINT
0B 6D 10          0        /"UD AT LINE # ".

          /ENTER SEARCH WITH D&E POINTING TO THE FIRST CHARACTER
          /IN THE TEXT BUFFER AND H&L POINTING TO THE BEGINNING
          /OF THE SYMBOLIC ADDRESS TABLE OR THE MNEMONIC TABLE.
          /C=THE SYMBOL TERMINATOR,B=# OF MEMORY LOCATIONS TO
          /SKIP AFTER THE TERMINATOR HAS BEEN FOUND, TO GET TO
          /THE NEXT "SYMBOL".

0B 6E EB  SEARCH, XCHG     /D&E SYMADD, H&L TEXT
0B 6F 22          SHLD
0B 70 45          TMPADD
0B 71 18          0
0B 72 2A  SKPMOR, LHLD     /GET THE ADDRESS OF THE BEGINNING
0B 73 45          TMPADD   /OF THE STRING BEING SEARCHED FOR
0B 74 18          0        /THIS IS THE ADDRESS IN THE TEXT BUFFER
0B 75 1A          LDAXD    /GET A CHARACTER FROM THE TABLE
0B 76 FE          CPI
0B 77 9B          BTERM
0B 78 C8          RZ       /WE EXIT THIS WAY THE FIRST TIME THROUGH
0B 79 1A  SRCH1,  LDAXD    /GET A CHARACTER FROM THE TABLE
0B 7A B9          CMPC     /IS IT A TERMINATOR (= OR ,) ?
0B 7B CA          JZ       /YES, WE FOUND THE END OF THE STRING
0B 7C 97          STREND   /IN THE TABLE, IS IT ALSO THE END OF
0B 7D 0B          0        /THE STRING IN THE TEXT BUFFER ?
0B 7E BE          CMPM     /TABLE CHARACTER = TEXT CHARACTER ?
0B 7F C2          JNZ      /NO, FIND THE NEXT ENTRY IN THE TABLE
0B 80 87          NXTMNE   /AND START THE COMPARISON AGAIN
0B 81 0B          0
```

```
0B 82 23          INXH     /YES, THEY MATCH
0B 83 13          INXD     /INCREMENT BOTH DATA POINTERS
0B 84 C3          JMP
0B 85 79          SRCH1
0B 86 0B          0
0B 87 13  NXTMNE, INXD     /THEY DIDN'T MATCH, FIND THE NEXT
0B 88 1A          LDAXD    /ENTRY IN THE TABLE BEING SEARCHED
0B 89 B9          CMPC     /FIND THE , OR = SIGN YET ?
0B 8A C2          JNZ
0B 8B 87          NXTMNE   /NO
0B 8C 0B          0
0B 8D C5  NOTPOM, PUSHB    /YES
0B 8E 13  SKIP,   INXD
0B 8F 05          DCRB
0B 90 C2          JNZ      /SKIP OVER THE 1 OR 2 OCTAL #'S DUE TO
0B 91 8E          SKIP     /THE MNEMONIC OR SYMBOLIC ADDRESS VALUE
0B 92 0B          0
0B 93 C1          POPB
0B 94 C3          JMP
0B 95 72          SKPMOR   /TRY IT AGAIN WITH A DIFFERENT ENTRY
0B 96 0B          0        /IN THE TABLE BEING SEARCHED

0B 97 13  STREND, INXD
0B 98 1A          LDAXD    /GET PAST = AND GET THE VALUE
0B 99 32          STA      /THIS IS FOR MOVES AND MATH, LOGIC
0B 9A 36          MNEVAL
0B 9B 18          0
0B 9C 13          INXD
0B 9D 1A          LDAXD
0B 9E 32          STA
0B 9F 37          MNEVAL+1
0B A0 18          0
0B A1 1B          DCXD     /FOR THE 2ND BYTE OF ADDRESS
0B A2 1B          DCXD     /IF A SYMBOLIC ADDRESS SEARCH
0B A3 22          SHLD
0B A4 43          REG      /SAVE H&L IN "REG" FOR PARTIAL MNEMONICS
0B A5 18          0
0B A6 C3          JMP
0B A7 B0          MNEND
0B A8 0B          0
0B A9 13  SAVVAL, INXD     /SKIP OVER THE , OR = IN THE TABLE
0B AA 1A          LDAXD    /GET THE TABLE VALUE
0B AB 32          STA
0B AC 36          MNEVAL   /SAVE IT
0B AD 18          0
0B AE AF          XRAA     /ON EXIT IF A=000, WE HAD A MATCH
0B AF C9          RET

          /"MNEND" CHECKS TO SEE IF THERE IS A VALID TERMINATOR
          /AFTER THE MNEMONIC.  A TERMINATOR COULD BE A CARRAGE
          /RETURN, SPACE, TAB, SLASH FOR A COMMENT (OR A COMMA
          /WHEN WE ARE ENTERING A SYMBOLIC ADDRESS INTO "SYMADD").

0B B0 7E  MNEND,  MOVAM
```

```
0B B1 FE          CPI
0B B2 0D          CR       /CR AFTER A MNEMONIC?
0B B3 CA          JZ
0B B4 A9          SAVVAL
0B B5 0B          0
0B B6 FE          CPI
0B B7 20          040      /SPACE AFTER THE MNEMONIC ?
0B B8 CA          JZ
0B B9 A9          SAVVAL
0B BA 0B          0
0B BB FE          CPI
0B BC 2F          057      /COMMENT AFTER MNEMONIC
0B BD CA          JZ
0B BE A9          SAVVAL
0B BF 0B          0
0B C0 FE          CPI
0B C1 0A          012      /A TAB OR THE # OF SPACES ?
0B C2 DA          JC
0B C3 A9          SAVVAL
0B C4 0B          0
0B C5 FE          CPI
0B C6 2C          054      /THE COMMA OF A SYMBOLIC ADDRESS
0B C7 CA          JZ
0B C8 A9          SAVVAL
0B C9 0B          0
0B CA FE          CPI      /IS THERE A + OR - AFTER
0B CB 2B          "+"      /THE SYMBOLIC ADDRESS
0B CC CA          JZ
0B CD D4          PORM
0B CE 0B          0
0B CF FE          CPI
0B D0 2D          "-"
0B D1 C2          JNZ
0B D2 8D          NOTPOM   /NOT A VALID TERMINATOR.  SKIP IT IN THE
0B D3 0B          0        /TABLE AND TRY THE NEXT ENTRY
0B D4 E5  PORM,   PUSHH    /CALCULATE THE OFFSET
0B D5 CD          CALL     /SAVE THE TEXT ADDRESS AND
0B D6 27          THRNMB   /CALCULATE THE OFFSET
0B D7 0D          0
0B D8 CA          JZ       /INVALID NUMBERS WERE USED
0B D9 F6          BDBIAS
0B DA 0B          0
0B DB 06          MVIB     /# IS IN C, SO SET B=000
0B DC 00          000
0B DD E1          POPH     /GET THE TEXT POINTER BACK
0B DE 7E          MOVAM    /SHOULD WE ADD OR SUBTRACT THE BIAS ?
0B DF FE          CPI
0B E0 2D          "-"
0B E1 CA          JZ       /SUBTRACT IT, IT WAS A "-"
0B E2 ED          SUBIAS
0B E3 0B          0
0B E4 2A  ADBIAS, LHLD     /GET THE BASE ADDRESS
0B E5 36          MNEVAL
0B E6 18          0
```

```
0B E7 09          DADB     /ADD THE OFFSET
0B E8 22  SAVBIS, SHLD     /AND SAVE THE NEW ADDRESS
0B E9 36          MNEVAL
0B EA 18          0
0B EB AF          XRAA     /NOW MAKE IT LOOK LIKE WE ACTUALLY
0B EC C9          RET      /FOUND THE ENTRY IN THE TABLE

0B ED 79  SUBIAS, MOVAC    /SUBTRACTION IS THE SAME AS ADDING
0B EE 2F          CMA      /THE 2'S COMPLEMENT
0B EF 4F          MOVCA
0B F0 06          MVIB
0B F1 FF          377
0B F2 03          INXB     /NOW WE HAVE INCREMENTED THE COMPLEMENT
0B F3 C3          JMP
0B F4 E4          ADBIAS
0B F5 0B          0

0B F6 E1  BDBIAS, POPH     /POP THE TEXT BUFFER ADDRESS FOR + OR -
0B F7 CD          CALL     /THE BIAS FACTOR DID NOT CONTAIN
0B F8 2C          UDOUT    /VALID NUMBERS, PRINT THE "UD"
0B F9 10          0        /ERROR MESSAGE, UNDEFINED SYMBOL
0B FA 2E          MVIL
0B FB 00          000      /SET THE ADDRESS TO 000 000
0B FC 26          MVIH
0B FD 00          000
0B FE C3          JMP
0B FF E8          SAVBIS
0C 00 0B          0

          /THIS IS USED FOR THE TTY OUTPUT ON THE 3RD PASS
          /IT TYPES OUT THE CURRENT ADDRESS COUNTER AS A
          /H & L ADDRESS AND THE VALUE OF THE MNEMONIC
          /AT THAT ADDRESS.

0C 01 2A  HLMOUT, LHLD     /GET THE "CURRENT ADDRESS CNTR" INTO H&L
0C 02 34          ADDCNT
0C 03 18          0
0C 04 CD          CALL     /PRINT OUT THE CONTENTS OF H&L
0C 05 AD          HLOUT
0C 06 05          0
0C 07 3A          LDA      /GET THE VALUE OF THE MNEMONIC
0C 08 36          MNEVAL   /AT THAT ADDRESS
0C 09 18          0
0C 0A CD          CALL     /AND PRINT IT OUT
0C 0B B2          OCTOUT
0C 0C 05          0
0C 0D 3E          MVIA     /AND PRINT AN ADDITIONAL SPACE AFTER IT
0C 0E A0          240
0C 0F C3          JMP
0C 10 0F          100
0C 11 18          0

          /"IOCHK" CHECKS WHICH PASS IS CURRENTLY BEING EXECUTED.
          /IF IT'S PASS 1 (PUTTING SYMB. ADDRESSES INTO "SYMADD")
```

```
          /"IOCHK" DOES NOTHING. IF ITS PASS 2, WE PUNCH A
          /PAPER TAPE (IF PASS 2, "PASS" = 120 (ASCII "P")). IF
          /PASS 3, WE GENERATE A PRINT OUT OF THE PROGRAM
          /("PASS" = 114 = ASCII "L" FOR LIST).

0C 12 3A  IOCHK,  LDA
0C 13 42          PASS
0C 14 18          0
0C 15 FE          CPI
0C 16 FF          377      /PASS1
0C 17 C8          RZ
0C 18 FE          CPI
0C 19 50          120      /P=PUNCH
0C 1A CA          JZ
0C 1B A9          BYTE
0C 1C 0F          0
0C 1D FE          CPI
0C 1E 4C          114
0C 1F C0          RNZ
0C 20 CD  LISTER, CALL     /PRINT THE HI AND LO MEMORY
0C 21 01          HLMOUT   /ADDRESS AND THE CONTENTS OF THAT ADDRESS
0C 22 0C          0
0C 23 2A  SPECL,  LHLD     /H&L NOW POINT TO THE FIRST CHARACTER
0C 24 2E          TFCLIN   /WITHIN A LINE OF TEXT
0C 25 18          0
0C 26 0E  SETTAB, MVIC     /NOW SET THE TAB STOP COUNTER TO 8
0C 27 08          010
0C 28 7E  LISTNT, MOVAM    /NOW GET A TEXT CHARACTER AND SEE
0C 29 FE          CPI      /WHAT WE SHOULD PRINT
0C 2A 0A          012      /0-10 OR 011 MEANS TO TYPE SPACES
0C 2B DA          JC       /UNTIL WE GET TO THE TAB STOP (C)
0C 2C 40          TABOK
0C 2D 0C          0
0C 2E FE          CPI
0C 2F 0D          CR       /FOUND A CR, TYPE A CR & LF AND
0C 30 CA          JZ       /THEN SEE WHETHER WE START A NEW PAGE
0C 31 49          CRFND
0C 32 0C          0
0C 33 CD          CALL     /NEITHER OF THOSE, SO JUST PRINT IT
0C 34 0F          IOO      /I-O DEVICE, OUTPUT TYPE
0C 35 18          0
0C 36 0D          DCRC     /DECREMENT THE TAB STOP COUNTER
0C 37 C2          JNZ      /=0 YET ?, NO GET ANOTHER CHARACTER
0C 38 3C          REENTR
0C 39 0C          0
0C 3A 0E          MVIC     /YES, RESET IT TO 8 (DECIMAL) THEN
0C 3B 08          010
0C 3C 23  REENTR, INXH     /NOW INCREMENT THE TEXT POINTER
0C 3D C3          JMP
0C 3E 28          LISTNT   /AND PRINT IT OUT ALSO
0C 3F 0C          0
```

```
0C 40 3E  TABOK,  MVIA    /IT WAS A TAB, PRINT SPACES TO THE
0C 41 A0          240     /NEXT TAB STOP.
0C 42 CD          CALL
0C 43 88          SIXSPC+4
0C 44 0F          0
0C 45 23          INXH
0C 46 C3          JMP
0C 47 26          SETTAB
0C 48 0C          0

0C 49 CD  CRFND,  CALL    /PRINT THE CRALF, THEN CHECK TO
0C 4A 39          CRLF
0C 4B 02          0
0C 4C C3          JMP     /SEE IF WE SHOULD START A NEW PAGE
0C 4D 33          PG
0C 4E 0F          0

0C 4F 21  READY,  LXIH    /SET THE BLOCK # TO 01
0C 50 2C          THOU+1
0C 51 18          0
0C 52 36          MVIM
0C 53 B1          261     /SET TEN THOU (UNITS OF BLK. #) TO 1.
0C 54 23          INXH
0C 55 36          MVIM
0C 56 B0          260     /SET HUN. THOU. (TENS OF BLK #) TO 0.
0C 57 CD          CALL
0C 58 08          LINE1   /SET THE LINE COUNTER TO 0001
0C 59 06          0
0C 5A 3A          LDA
0C 5B 3D          MORT
0C 5C 18          0
0C 5D FE          CPI
0C 5E 4D          "M"     /IS THE PROGRAM IN MEMORY ?
0C 5F C8          RZ      /YES DON'T TYPE OUT MULTIPLE R'S
0C 60 CD  NOR,    CALL
0C 61 39          CRLF    /TYPE A CR, LF
0C 62 02          0
0C 63 21          LXIH
0C 64 8E          RDY     /THEN TYPE "R ?"
0C 65 11          0
0C 66 CD          CALL
0C 67 7A          TOUT
0C 68 05          0
0C 69 CD  NOR1,   CALL    /GET AN INPUT CHAR. FROM A TTY OR CRT
0C 6A 06          IOECHO
0C 6B 18          0
0C 6C FE          CPI
0C 6D 03          CTRLC   /CTRL-C?
0C 6E CA          JZ      /YES, GO BACK TO THE COMMAND DECODER
0C 6F EF          NOGOOD
0C 70 03          0
0C 71 FE          CPI
0C 72 00          000     /THERE MIGHT BE TRAILER IN THE READER
0C 73 CA          JZ      /FROM THE PREVIOUS PASS, SO IGNORE IT.
```

```
0C 74 69          NORI    /(ONLY IF YOU USE A TTY WILL THIS BE TRUE)
0C 75 0C          0
0C 76 FE          CPI
0C 77 52          "R"     /R = READ OR READY
0C 78 C2          JNZ
0C 79 60          NOR     /WASN'T A 000 OR AN R, TRY AGAIN
0C 7A 0C          0
0C 7B 2A  RDLINE, LHLD
0C 7C 22          TXTBUF
0C 7D 18          0
0C 7E CD  RDRMOR, CALL
0C 7F 0C          IOR     /I-O DEVICE, READER TYPE
0C 80 18          0
0C 81 FE          CPI
0C 82 03          CTRLC
0C 83 C2          JNZ     /A CNTRL-C AT THE END OF A TAPE RESETS
0C 84 A5          NOTCC   /THE LINE COUNTER FOR ERROR MESSAGES.
0C 85 0C          0
0C 86 E5          PUSHH   /SAVE THE 1 LINE STORAGE POINTER
0C 87 CD          CALL
0C 88 08          LINE1   /SET THE LINE COUNTER TO 0001
0C 89 06          0
0C 8A 21          LXIH    /NOW INCREMENT THE BLOCK COUNTER
0C 8B 2C          THOU+1
0C 8C 18          0
0C 8D 0E          MVIC    /2 DIGIT BLOCK NUMBER
0C 8E 02          002
0C 8F CD          CALL    /NOW INCREMENT THE NUMBER
0C 90 F9          NXTNBM
0C 91 05          0
0C 92 E1          POPH    /RECOVER THE STORAGE POINTER
0C 93 AF          XRAA    /NEW PAPER TAPE BLOCK, RESET
0C 94 32          STA     /THE PAGE NUMBER
0C 95 47          PAGNMB
0C 96 18          0
0C 97 3A          LDA     /NOW SEE IF WE SHOULD START A
0C 98 42          PASS    /NEW PAGE.  WE SHOULD ONLY ON
0C 99 18          0       /THE THIRD PASS.
0C 9A FE          CPI
0C 9B 4C          "L"     /L=LISTING, PASS 3
0C 9C C2          JNZ
0C 9D A2          NOPAGE  /NOT PASS3, DON'T ADVANCE THE PAPER
0C 9E 0C          0
0C 9F CD          CALL    /OK, FINISH UP THE
0C A0 FD          FINPAG  /CURRENT PAGE
0C A1 0E          0
0C A2 C3  NOPAGE, JMP
0C A3 7E          RDRMOR  /AND GET THE NEXT PAPER TAPE CHARACTER
0C A4 0C          0
0C A5 FE  NOTCC,  CPI
0C A6 00          000     /7BIT LDR
0C A7 CA          JZ
0C A8 7E          RDRMOR
0C A9 0C          0
```

```
0C AA FE          CPI
0C AB 7F          177      /7 BIT RUBOUT
0C AC CA          JZ
0C AD 7E          RDRMOR
0C AE 0C          0
0C AF FE          CPI
0C B0 24          044      /THE DOLLAR SIGN TERMINATES INPUT
0C B1 CA          JZ
0C B2 C1          ENDTXT
0C B3 0C          0
0C B4 FE          CPI
0C B5 0A          LF       /LINE FEED?
0C B6 CA          JZ       /YES, THEN IGNORE IT AND GET
0C B7 7E          RDRMOR   /ANOTHER TAPE CHARACTER
0C B8 0C          0
0C B9 77          MOVMA    /SAVE THE CHARACTER IN MEMORY
0C BA FE          CPI
0C BB 0D          CR       /WAS IT A CR?, YES, THEN ITS THE
0C BC C8          RZ       /END OF 1 LINE, NOW INTERPRETE IT.
0C BD 23          INXH     /NONE OF THE ABOVE, INCREMENT H&L
0C BE C3          JMP      /AND THEN READ ANOTHER CHARACTER
0C BF 7E          RDRMOR
0C C0 0C          0
0C C1 36  ENDTXT, MVIM     /END OF THE PAPER TAPE
0C C2 9B          BTERM    /SO SAVE A 233 TERMINATOR AT THE
0C C3 C9          RET      /END OF THE LINE AND RETURN

          /THIS ROUTINE INTERPRETES THE TWO ADDRESS
          /BYTES SEPARATED BY A SPACE WHEN WE HAVE
          /A DEFINED ADDRESS WITH A * OR WHEN WE
          /USE THE PSEUDO-OP "DW".

0C C4 CD  ADBYTE, CALL
0C C5 27          THRNMB   /CALCULATE THE HI ADDRESS
0C C6 0D          0
0C C7 CA          JZ       /NOT A VALID NUMBER, SO
0C C8 06          NOGOAD   /PRINT AN ERROR MESSAGE
0C C9 0D          0
0C CA 59          MOVEC    /SAVE THE HI ADDRESS IN "E"
0C CB 23          INXH     /INCREMENT THE TEXT BUFFER POINTER
0C CC 7E          MOVAM    /CHECK FOR A SPACE SEPARATING THE #'S
0C CD FE          CPI
0C CE 20          040      /2 #'S, H&L, SEPARATED BY A SPACE
0C CF C2          JNZ
0C D0 06          NOGOAD   /THERE WAS NO SPACE,
0C D1 0D          0        /SO PRINT AN ERROR MESSAGE
0C D2 CD          CALL
0C D3 27          THRNMB   /THERE WAS A SPACE, CALCULATE THE LO ADD.
0C D4 0D          0
0C D5 CA          JZ
0C D6 06          NOGOAD   /NOT A VALID #, SO PRINT AN ERROR MESSAGE
0C D7 0D          0
0C D8 53          MOVDE    /SAVE THE HI ADDRESS IN "D"
0C D9 59          MOVEC    /SAVE THE LO ADDRESS IN "E"
```

```
0C DA EB          XCHG    /PUT THE CALCULATED ADDRESS INTO H&L
0C DB C9          RET     /AND RETURN WITH IT IN H&L

0C DC CD  ADDOK,  CALL    /CALCULATE THE ADDRESS
0C DD C4          ADBYTE
0C DE 0C          0
0C DF 3E  FINADD, MVIA    /AN ADDRESS HAS BEEN DEFINED, SO
0C E0 FF          377     /SET THE "ADDRESS INDICATOR REGISTER" TO
0C E1 32          STA     /377.
0C E2 16          ADCHK
0C E3 18          0
0C E4 22          SHLD    /AND THEN SAVE THE ADDRESS IN THE
0C E5 34          ADDCNT  /"CURRENT ADDRESS COUNTER"
0C E6 18          0
0C E7 EB          XCHG    /PUT THE CURRENT ADDRESS IN D&E
0C E8 3A          LDA     /DETERMINE WHICH PASS THIS IS
0C E9 42          PASS
0C EA 18          0
0C EB FE          CPI
0C EC 50          "P"     /P=PUNCH=PASS2, PUNCHING A TAPE
0C ED C2          JNZ     /NOT PASS2, ITS PASS 1 OR 3
0C EE F6          NTADD
0C EF 0C          0
0C F0 CD          CALL    /ITS PASS 2, PUNCH THE ADDRESS
0C F1 17          TAPADD  /IDENTIFIER (CHANNEL 7) ON THE TAPE
0C F2 0D          0       /AND THE CURRENT ADDRESS
0C F3 C3          JMP
0C F4 01          IODONE  /THEN FIND THE NEXT LINE OF TEXT
0C F5 0D          0
0C F6 FE  NTADD,  CPI
0C F7 4C          "L"     /L=LIST=PASS 3
0C F8 C2          JNZ
0C F9 01          IODONE  /NOT PASS 3, MUST BE PASS 1, KEEP GOING
0C FA 0D          0
0C FB CD          CALL    /ITS PASS 3, TYPE SOME SPACES
0C FC 1B          SPCIT
0C FD 10          0
0C FE CD          CALL    /THEN TYPE THE ADDRESS
0C FF 23          SPECL
0D 00 0C          0
0D 01 06  IODONE, MVIB    /NOW FIND THE NEXT LINE OF TEXT
0D 02 00          000
0D 03 C3          JMP
0D 04 A5          FINDCR
0D 05 0D          0

0D 06 CD  NOGOAD, CALL
0D 07 0E          ENOUT   /TYPE "EN" ERROR MESSAGE
0D 08 0D          0
0D 09 2E          MVIL
0D 0A 00          000     /SET THE H&L ADDRESS TO 000 000
0D 0B 26          MVIH
0D 0C 00          000
0D 0D C9          RET     /AND THEN RETURN FROM "ADBYTE"
```

```
0D 0E E5  BNOUT,  PUSHH
0D 0F 21          LXIH    /THIS IS TO TYPE THE "BN"
0D 10 6F          BN      /(BAD NUMBER) ERROR MESSAGE
0D 11 11          0
0D 12 CD          CALL    /TYPE "BN" PLUS THE LINE NUMBER
0D 13 0E          LIST1   /WHERE THE ERROR OCCURRED.
0D 14 0F          0
0D 15 E1          POPH    /GET THE POINTER BACK
0D 16 C9          RET

0D 17 3E  TAPADD, MVIA    /PUNCH CHANNEL 7 FOR THE ADDRESS
0D 18 40          100     /IDENTIFIER (FLAG)
0D 19 CD          CALL
0D 1A 09          IOP     /I-O DEVICE, PUNCH TYPE
0D 1B 18          0
0D 1C 2A          LHLD    /THEN GET THE CONTENTS OF THE
0D 1D 34          ADDCNT  /"CURRENT ADDRESS COUNTER"
0D 1E 18          0
0D 1F 7C  RADD1,  MOVAH   /AND PUNCH THE ADDRESS OUT
0D 20 CD          CALL    /ON THE TAPE AS 4 FRAMES
0D 21 E4          BYTE1
0D 22 0F          0
0D 23 7D          MOVAL
0D 24 C3          JMP
0D 25 E4          BYTE1
0D 26 0F          0
```

```
            /THIS SUBROUTINE CONVERTS THE ASCII CHARACTERS STORED
            /IN THE TEXT BUFFER TO EIGHT-BIT NUMBERS.  EITHER
            /OCTAL OR HEXADECIMAL NUMBERS CAN BE CONVERTED.

            /AFTER THE CALL TO "THRNMB", A JZ TO AN ERROR
            /ROUTINE CAN BE PERFORMED, WHICH MEANS THAT THE
            /NUMBER COULD NOT BE PROPERLY CONVERTED.

0D 27 22  THRNMB, SHLD    /SAVE THE TEXT BUFFER POINTER
0D 28 26          TMP     /THAT ADDRESSES THE FIRST DIGIT.
0D 29 18          0
0D 2A 16          MVID
0D 2B 02          002     /2 HEX DIGITS PER 8 BIT WORD
0D 2C AF          XRAA
0D 2D 4F          MOVCA   /THE RESULT WILL BE SAVED IN C.
0D 2E 23  CONVHX, INXH
0D 2F 7E          MOVAM   /GET A CHARACTER
0D 30 FE          CPI
0D 31 30          "0"
0D 32 DA          JC      /ITS NOT HEX, IS IT OCTAL ?
0D 33 60          TRYOCT
0D 34 0D          0
0D 35 FE          CPI     /LESS THAN ASCII :?
0D 36 3A          ":"
0D 37 DA          JC
0D 38 46          HEXNMB  /OK, ITS 0-9, CONVERT IT
0D 39 0D          0
0D 3A FE          CPI
0D 3B 41          "A"     /LESS THAN ASCII "A" ?
0D 3C DA          JC
0D 3D 60          TRYOCT  /SEE IF ITS OCTAL INSTEAD
0D 3E 0D          0
0D 3F FE          CPI
0D 40 47          "G"     /F IS THE HIGHEST VALID HEX CHARACTER
0D 41 D2          JNC
0D 42 60          TRYOCT  /TRY INTERPRETING IT AS OCTAL
0D 43 0D          0
0D 44 C6          ADI     /TO GET HERE, ITS A-F
0D 45 09          011
0D 46 E6  HEXNMB, ANI     /SAVE THE FOUR LSBS
0D 47 0F          017
0D 48 47          MOVBA   /SAVE THE VALUE IN B
0D 49 79          MOVAC   /GET THE PREVIOUS RESULT
0D 4A 07          RLC     /MULTIPLY IT BY 16
0D 4B 07          RLC
0D 4C 07          RLC
0D 4D 07          RLC
0D 4E 80          ADDB    /ADD THE VALUE (DIGIT) JUST FOUND
0D 4F 4F          MOVCA   /SAVE THE RESULT
0D 50 15          DCRD    /2 DIGITS YET ?
0D 51 C2          JNZ     /NO, CONVERT ANOTHER ONE
0D 52 2E          CONVHX
0D 53 0D          0
0D 54 23          INXH    /YES, SEE IF THERE IS AN "H"
```

```
0D 55 7E          MOVAM   /AFTER THE TWO DIGIT HEX NUMBER
0D 56 FE          CPI
0D 57 48          "H"     /H AFTER THE 2 DIGITS ?
0D 58 C2          JNZ
0D 59 60          TRYOCT  /NO "H", IS IT OCTAL ?
0D 5A 0D          0
0D 5B 23          INXH    /INCREMENT PAST THE H
0D 5C 7E          MOVAM   /THIS MUST BE A TERMINATOR
0D 5D C3          JMP
0D 5E 82          NONMB
0D 5F 0D          0

0D 60 2A  TRYOCT, LHLD    /GET THE TEXT BUFFER POINTER
0D 61 26          TMP     /THAT POINTS TO THE STRING
0D 62 18          0       /OF CHARACTERS
0D 63 16          MVID
0D 64 03          003     /DIGIT COUNTER
0D 65 AF          XRAA
0D 66 4F          MOVCA
0D 67 23  NXTNMB, INXH    /FIRST WE WILL TRY TO CONVERT THE
0D 68 7E          MOVAM   /NUMBERS ASSUMING OCTAL DIGITS
0D 69 FE          CPI
0D 6A 30          "0"
0D 6B DA          JC
0D 6C 82          NONMB   /OK, SEE IF THERE'S A VALID TERMINATOR
0D 6D 0D          0
0D 6E FE          CPI
0D 6F 38          "8"
0D 70 D2          JNC
0D 71 82          NONMB   /OK, SEE IF THERE'S A VALID TERMINATOR
0D 72 0D          0
0D 73 E6          ANI     /ITS AN OCTAL NUMBER, SAVE THE
0D 74 07          007     /THREE LSBS
0D 75 47          MOVBA   /SAVE THE VALUE
0D 76 79          MOVAC   /GET THE PREVIOUS VALUE
0D 77 07          RLC     /MULTIPLY IT BY 8
0D 78 07          RLC
0D 79 07          RLC
0D 7A 80          ADDB    /ADD THE VALUE JUST ENTERED
0D 7B 4F          MOVCA   /AND SAVE THE RESULT
0D 7C 15          DCRD    /DECREMENT THE DIGIT COUNT
0D 7D C2          JNZ     /IF ITS NOT ZERO, GET ANOTHER
0D 7E 67          NXTNMB  /CHARACTER
0D 7F 0D          0
0D 80 23          INXH    /WE HAD THREE DIGITS, SEE IF
0D 81 7E          MOVAM   /THERE'S A TERMINATOR AFTER THEM
0D 82 2B  NONMB,  DCXH
0D 83 FE          CPI     /IS THERE A SPACE AFTER THE #?
0D 84 20          040
0D 85 CA          JZ      /YES, IT'S A VALID TERMINATOR
0D 86 97          NOAI
0D 87 0D          0
0D 88 FE          CPI     /IS THERE A CARRIAGE RETURN
0D 89 0D          CR      /AFTER THE NUMBER?
```

```
0D 8A CA          JZ      /YES, THATS A VALID TERMINATOR
0D 8B 97          NOA1
0D 8C 0D          0
0D 8D FE          CPI     /IS THERE A LINE FEED OR TAB
0D 8E 0A          LF      /AFTER THE NUMBER?
0D 8F DA          JC      /YES, THEY ARE VALID TERMINATORS
0D 90 97          NOA1
0D 91 0D          0
0D 92 FE          CPI     /IS THERE A SLASH (/) FOR A COMMENT)
0D 93 2F          057     /AFTER THE NUMBER?
0D 94 C2          JNZ     /NO, THEN IT IS NOT TERMINATED
0D 95 9B          BADNMB  /PROPERLY, ITS A BAD NUMBER
0D 96 0D          0
0D 97 7A  NOA1,   MOVAD   /TERMINATED OK, ANY #S ENTERED?
0D 98 FE          CPI
0D 99 03          003
0D 9A C9          RET
0D 9B AF  BADNMB, XRAA    /WE HAD AN ERROR
0D 9C 4F          MOVCA   /CLEAR "C" TO 000
0D 9D FE          CPI
0D 9E 00          000     /SET THE ZERO FLAG
0D 9F C9          RET
```

```
            /THE B REGISTER IS USED TO SENSE WHETHER OR NOT
            /TO INCREMENT "ADDCNT" THE CURRENT ADDRESS COUNTER
            /AND INCREMENT THE "LINCNT" USED FOR ERROR MESSAGES.
            /"LINCNT" IS ALWAYS INCREMENTED, BUT IF A LINE
            /OF TEXT DOES NOT CONTAIN DATA OR A MNEMONIC,
            /B WILL = 000, AND "ADDCNT" WILL NOT BE INCREMENTED.

0D A0 C5  CRIO,    PUSHB   /SAVE THE "B" FLAG
0D A1 CD           CALL    /CHECK THE PASS 0, AND IF ITS 2 OR 3
0D A2 12           IOCHK   /PUNCH OR LIST THE PROGRAM
0D A3 0C           0       /("IOCHK" = I-O CHECK)
0D A4 C1           POPB    /GET THE "B" FLAG BACK
0D A5 78  FINDCR,  MOVAB
0D A6 FE           CPI     /INCREMENT THE ADDRESS AND LINE COUNT ?
0D A7 00           000
0D A8 CA           JZ      /JUST INCREMENT THE LINE COUNT
0D A9 C7           NODATA  /(USED FOR THE ERROR MESSAGES)
0D AA 0D           0
0D AB 3A  ADDINC,  LDA     /WHAT PASS IS THIS?
0D AC 42           PASS
0D AD 18           0
0D AE FE           CPI     /PASS 1?
0D AF FF           377
0D B0 CA           JZ      /YES, THEN SIMPLY INCREMENT THE ADDRESS
0D B1 C0           AD2OR3  /(ADDCNT) AND LINE COUNT (LINCNT).
0D B2 0D           0
0D B3 3A           LDA     /ITS PASS 2 OR 3, GET THE ADDRESS
0D B4 16           ADCHK   /SET REGISTER.
0D B5 18           0
0D B6 B7           ORAA    /ZERO OR NON-ZERO?
0D B7 C2           JNZ     /OK, AN ADDRESS WAS SET BY AN *
0D B8 C0           AD2OR3  /SO ITS OK TO INCREMENT THE
0D B9 0D           0       /ADDRESS IN ADDCNT.
0D BA 21           LXIH    /WE CAN'T INCREMENT AN ADDRESS THAT
0D BB 78           NOADR   /WE DO NOT HAVE, SO PRINT
0D BC 11           0       /AN ERROR MESSAGE "NA AT LINE 0."
0D BD C3           JMP     /THIS IS A FATAL ERROR.
0D BE 26           FATAL
0D BF 0B           0
0D C0 2A  AD2OR3,  LHLD
0D C1 34           ADDCNT  /GET, INCREMENT BY 1 AND STORE
0D C2 18           0       /THE "CURRENT ADDRESS COUNTER"
0D C3 23           INXH
0D C4 22           SHLD
0D C5 34           ADDCNT
0D C6 18           0
0D C7 CD  NODATA,  CALL
0D C8 F4           LINC    /INCREMENT THE DECIMAL LINE 0
0D C9 05           0       /BY 1.
0D CA 3A  OKINC,   LDA     /IF WE ARE ASSEMBLING THE SOURCE
0D CB 3D           MORT    /FROM TAPE, WE SHOULD NOW GET
0D CC 18           0       /THE NEXT LINE OF SOURCE.
0D CD FE           CPI
```

```
0D CE 54            "T"
0D CF CA            JZ       /SOURCE IS FROM TAPE, GET
0D D0 E2            READIT   /THE NEXT LINE
0D D1 0D            0
0D D2 2A            LHLD     /IT IS NOT FROM TAPE, SO FIND
0D D3 2E            TFCLIN   /THE BEGINNING OF THE NEXT LINE
0D D4 18            0        /OT TEXT STORED IN MEMORY.
0D D5 7E  NCRYET,   MOVAM    /GET A TEXT CHARACTER
0D D6 23            INXH
0D D7 FE            CPI      /IS IT THE CR AT THE END OF THE LINE ?
0D D8 0D            CR
0D D9 C2            JNZ      /NO, KEEP LOOKING FOR THE CR
0D DA D5            NCRYET
0D DB 0D            0
0D DC 22  SETTFC,   SHLD     /YES, SAVE THE ADDRESS OF THE FIRST
0D DD 2E            TFCLIN   /CHARACTER IN THE NEXT LINE.
0D DE 18            0
0D DF 06            MVIB     /RESET THE FLAG REGISTER TO 0
0D E0 00            000
0D E1 C9            RET

0D E2 CD  READIT,   CALL     /SOURCE BEING ASSEMBLED IS FROM TAPE
0D E3 7B            RDLINE   /SO READ THE NEXT LINE OF TEXT
0D E4 0C            0
0D E5 2A            LHLD     /H&L POINT TO THE BEGINNING
0D E6 22            TXTBUF   /OF THE LINE
0D E7 18            0
0D E8 C3            JMP      /SO SAVE H&L IN "TFCLIN"
0D E9 DC            SETTFC
0D EA 0D            0

          /WHEN ITS TIME AT THE END OF THE THIRD PASS TO PRINT
          /OUT THE CONTENTS OF THE SYMBOL TABLE, WE COME HERE.
          /FIRST, THE SYMBOLIC ADDRESSES ARE SORTED USING A
          /BUBBLE SORT ALGORITHM.  THEN THE SYMBOLIC ADDRESSES
          /ARE PRINTED IN FOUR COLUMNS, ALONG WITH THEIR ASSIGNED
          /ADDRESSES.

0D EB 3E  SYMOUT,   MVIA
0D EC 04            004      /# OF COLUMNS
0D ED 32            STA
0D EE 36            MNEVAL   /"MNEVAL" IS JUST USED
0D EF 18            0        /AS A STORAGE LOCATION (TEMPORARY)
0D F0 CD            CALL     /FINISH UP THE CURRENT PAGE
0D F1 FD            FINPAG   /BY TYPING ENOUGH CR'S AND
0D F2 0E            0        /LF'S TO GET TO THE NEXT PAGE
0D F3 3E  SORT,     MVIA     /SET THE "EXCHANGE INDICATOR" TO
0D F4 FF            377      /377.
0D F5 32            STA      /SAVE THE EXCHANGE INDICATOR.
0D F6 26            TMP      /IF AN EXCHANGE TAKES PLACE, THIS
0D F7 18            0        /NUMBER IS SET TO ZERO.
0D F8 2A            LHLD     /LOAD H&L WITH THE STARTING ADDRESS
0D F9 39            SYMADD   /OF THE SYMBOLIC ADDRESS TABLE.
0D FA 18            0
```

```
0D FB CD            CALL     /CALL THE BUBBLE SORT SUBROUTINE.
0D FC 5F            BSORT
0D FD 0E            0
0D FE 3A            LDA      /GET THE EXCHANGE INDICATOR
0D FF 26            TMP
0E 00 18            0
0E 01 FE            CPI      /ANY EXCHANGES TAKE PLACE?
0E 02 FF            377
0E 03 C2            JNZ      /YES, THEN MAKE ANOTHER PASS THROUGH
0E 04 F3            SORT     /THE SYMBOLIC ADDRESS TABLE WITH
0E 05 0D            0        /THE "BSORT" SUBROUTINE.
0E 06 2A            LHLD     /TEH TABLE IS SORTED, GET ITS
0E 07 39            SYMADD   /STARTING ADDRESS.
0E 08 18            0
0E 09 16  NOCRY,    MVID     /A MAXIMUM OF EIGHT CHARACTERS CAN
0E 0A 08            010      /BE PRINTED.
0E 0B 7E  TYPSYM,   MOVAM    /GET A CHARACTER FROM THE TABLE.
0E 0C FE            CPI      /IS IT THE EQUAL SIGN STORED AT THE
0E 0D 3D            075      /END OF EVERY SYMBOL?
0E 0E CA            JZ       /YES, THEN PRINT IT, FOLLOWED BY
0E 0F 2B            TYPEQ    /A 16-BIT OCTAL OR HEX ADDRESS
0E 10 0E            0
0E 11 FE            CPI      /REACHED THE END OF THE TABLE?
0E 12 9B            BTERM
0E 13 CA            JZ       /YES, THEN PRINT THE NUMBER OF ERRORS
0E 14 E5            LASTPG   /FROM THE THIRD PASS, AND ENOUGH CARRIAGE
0E 15 0E            0        /RETURNS AND LINE FEEDS TO FINISH THE PAGE.
0E 16 15            DCRD     /PRINTED EIGHT CHARACTERS YET?
0E 17 CA            JZ       /YES, WE HAVE, NOW FIND THE =
0E 18 21            NOEQY    /SIGN AND THEN PRINT THE
0E 19 0E            0        /16-BIT ADDRESS.
0E 1A CD            CALL     /PRINT THE CHARACTER READ
0E 1B 0F            IOO      /FROM THE TABLE
0E 1C 18            0
0E 1D 23            INXH     /INCREMENT THE TABLE ADDRESS
0E 1E C3            JMP      /AND READ THE NEXT CHARACTER.
0E 1F 0B            TYPSYM
0E 20 0E            0
0E 21 7E  NOEQY,    MOVAM    /THERE ARE OVER 8 CHARACTERS IN THE
0E 22 FE            CPI      /SYMBOL, SO WE ONLY TYPE THE FIRST
0E 23 3D            075      /8 AND THEN FIND THE ADDRESS
0E 24 CA            JZ       /WE FIRST FIND THE = BEFORE THE ADDRESS
0E 25 32            EOUT
0E 26 0E            0
0E 27 23            INXH     /KEEP LOOKING FOR THE =
0E 28 C3            JMP
0E 29 21            NOEQY    /NO = SIGN YET
0E 2A 0E            0

0E 2B 3E  TYPEQ,    MVIA     /PRINTED THE =, SO PRINT A SPACE.
0E 2C A0            240
0E 2D 4A            MOVCD    /THEN PRINT THE NUMBER OF SPACES
0E 2E 0D            DCRC     /SPECIFIED BY THE C REGISTER.
0E 2F C4            CNZ
```

```
0E 30 88          SIXSPC+4
0E 31 0F          0
0E 32 7E  EOUT,   MOVAM    /NOW GET THE = AND PRINT IT.
0E 33 CD          CALL
0E 34 0F          IOO      /I-O DEVICE, OUTPUT TYPE
0E 35 18          0
0E 36 23          INXH
0E 37 5E          MOVEM    /GET THE LO ADDRESS
0E 38 23          INXH
0E 39 7E          MOVAM    /THEN THE HI ADDRESS
0E 3A 23          INXH
0E 3B CD          CALL
0E 3C B2          OCTOUT   /PRINT THE HI ADDRESS
0E 3D 05          0
0E 3E 7B          MOVAE    /PUT THE LO ADDRESS IN "A"
0E 3F CD          CALL
0E 40 B2          OCTOUT   /THEN PRINT IT OUT ALSO
0E 41 05          0
0E 42 3E          MVIA     /A COUPLE OF SPACES TO THE NEXT SYMBOL
0E 43 A0          240
0E 44 CD          CALL
0E 45 0F          IOO      /I-O DEVICE, OUTPUT TYPE
0E 46 18          0
0E 47 3A          LDA
0E 48 36          MNEVAL   /GET THE "COLUMN COUNTER" ( WE HAVE
0E 49 18          0        /4 COULMNS OF SYMBOLIC ADDRESSES
0E 4A 3D          DCRA     /ACROSS A PAGE).  4 COLUMNS YET ?
0E 4B 32          STA
0E 4C 36          MNEVAL
0E 4D 18          0
0E 4E C2          JNZ      /NO, TYPE ANOTHER SYMBOL
0E 4F 09          NOCRY
0E 50 0E          0
0E 51 3E          MVIA     /YES, RESET THE COLUMN COUNTER
0E 52 04          004
0E 53 32          STA
0E 54 36          MNEVAL
0E 55 18          0
0E 56 CD          CALL     /THEN PRINT A CR & LF
0E 57 39          CRLF
0E 58 02          0
0E 59 CD          CALL
0E 5A 33          PG       /CHECK THE NUMBER OF LINES
0E 5B 0F          0        /ON THIS PAGE SO FAR
0E 5C C3          JMP
0E 5D 09          NOCRY
0E 5E 0E          0

0E 5F 22  BSORT,  SHLD     /SAVE THE STARTING ADDRESS OF
0E 60 34          ADDCNT   /THE SYMBOLIC ADDRESS TABLE
0E 61 18          0
0E 62 FE          CPI      /ARE WE AT THE END OF THE TABLE?
0E 63 9B          BTERM
0E 64 C8          RZ       /YES, THEN RETURN, NO SYMBOLS.
```

```
0E 65 23  BSORT1, INXH    /INCREMENT THE ADDRESS BY ONE.
0E 66 7E          MOVAM   /GET A CHARACTER
0E 67 FE          CPI     /FOUND THE EQUAL SIGN AT THE END
0E 68 3D          "="     /OF THE SYMBOL YET?
0E 69 C2          JNZ     /NO, THEN KEEP LOOKING
0E 6A 65          BSORT1
0E 6B 0E          0
0E 6C 23          INXH    /FOUND IT, INCREMENT PAST THE 16-BIT
0E 6D 23          INXH    /ADDRESS TO THE FIRST CHARACTER OF THE
0E 6E 23          INXH    /SECOND SYMBOLIC ADDRESS.
0E 6F 7E          MOVAM   /DOES THE SECOND SYMBOLIC ADDRESS
0E 70 FE          CPI     /EXIST?
0E 71 9B          BTERM
0E 72 C8          RZ      /ONLY 1 SYMBOL, ITS "SORTED", RETURN
0E 73 2A          LHLD    /GET THE ADDRESS OF THE FIRST
0E 74 34          ADDCNT
0E 75 18          0
0E 76 22  BSORT2, SHLD    /SAVE THE ADDRESS OF THE FIRST
0E 77 34          ADDCNT
0E 78 18          0
0E 79 7E  NOEQU,  MOVAM   /NOW FIND THE SECOND SYMBOLIC
0E 7A FE          CPI     /ADDRESS.
0E 7B 9B          BTERM
0E 7C C8          RZ
0E 7D 23          INXH
0E 7E FE          CPI
0E 7F 3D          "="
0E 80 C2          JNZ
0E 81 79          NOEQU
0E 82 0E          0
0E 83 23          INXH
0E 84 23          INXH
0E 85 22          SHLD    /OK, FOUND IT, SAVE ITS ADDRESS
0E 86 32          LINCNT
0E 87 18          0
0E 88 EB          XCHG    /ADDRESS OF 2ND IN D&E
0E 89 2A          LHLD
0E 8A 34          ADDCNT  /GET THE ADDRESS OF THE FIRST STRING
0E 8B 18          0       /BACK INTO H&L
0E 8C EB          XCHG    /1ST IN D&E, 2ND IN H&L
0E 8D 1A  NCHAR,  LDAXD   /GET A CHARACTER FROM THE 1ST
0E 8E FE          CPI
0E 8F 3D          "="     /END OF THE FIRST?
0E 90 C2          JNZ
0E 91 9A          NOTEND
0E 92 0E          0
0E 93 13          INXD    /FOUND THE END OF THE 1ST,
0E 94 13          INXD    /GO TO THE 2ND
0E 95 13          INXD
0E 96 EB          XCHG
0E 97 C3          JMP
0E 98 76          BSORT2
0E 99 0E          0
```

```
0E 9A 47  NOTEND, MOVBA   /SAVE 1ST IN B, SEE IF WE HAVE
0E 9B 7E          MOVAM   /GOT ONE OF THE 2ND CHARACTERS
0E 9C FE          CPI
0E 9D 9B          BTERM
0E 9E C8          RZ      /YES, THEN RETURN
0E 9F FE          CPI
0E A0 3D          "="     /1ST > 2ND ?
0E A1 CA          JZ      /YES, SO EXCHANGE THEM
0E A2 B0          SWAP
0E A3 0E          0
0E A4 B8          CMPB    /COMPARE 1ST IN B TO 2ND IN A
0E A5 DA          JC      /1ST IS GREATER THAN AND NOT EQUAL
0E A6 B0          SWAP    /TO 2ND, SO THEY MUST BE EXCHANGED
0E A7 0E          0
0E A8 C2          JNZ
0E A9 CD          FNXSYM
0E AA 0E          0
0E AB 23  SYMCEQ, INXH    /1ST = 2ND, SO EXAMINE THE NEXT
0E AC 13          INXD    /TWO CHARACTERS IN THE STRINGS
0E AD C3          JMP
0E AE 8D          NCHAR
0E AF 0E          0

0E B0 AF  SWAP,   XRAA
0E B1 32          STA
0E B2 26          TMP     /SET THE FLAG TO INDICATE THAT A EXCHANGE
0E B3 18          0       /HAS TAKEN PLACE
0E B4 2A          LHLD    /GET THE ADDRESS OF THE FIRST
0E B5 34          ADDCNT  /STRING
0E B6 18          0
0E B7 EB          XCHG
0E B8 21          LXIH
0E B9 A8          BUFF2
0E BA 18          0
0E BB CD          CALL    /MOVE THE 1ST STRING TO BUFF2
0E BC D3          MOVES
0E BD 0E          0
0E BE 2A          LHLD    /GET THE ADDRESS FOR THE 1ST
0E BF 34          ADDCNT  /STRING AGAIN
0E C0 18          0
0E C1 CD          CALL    /MOVE THE SECOND STRING TO WHERE
0E C2 D3          MOVES   /WHERE THE FIRST WAS STORED IN
0E C3 0E          0       /THE SYMBOL TABLE
0E C4 11          LXID
0E C5 A8          BUFF2
0E C6 18          0
0E C7 22          SHLD    /SAVE THE ADDRESS
0E C8 32          LINCNT
0E C9 18          0
0E CA CD          CALL
0E CB D3          MOVES   /MOVE THE FIRST FROM BUFF2 BACK
0E CC 0E          0       /INTO THE SYMBOL TABLE
0E CD 2A  FNXSYM, LHLD
0E CE 32          LINCNT
```

```
0E CF 18            0
0E D0 C3            JMP
0E D1 76            BSORT2
0E D2 0E            0

0E D3 1A  MOVES,    LDAXD
0E D4 77            MOVMA
0E D5 23            INXH
0E D6 13            INXD
0E D7 FE            CPI
0E D8 3D            "="
0E D9 C2            JNZ
0E DA D3            MOVES
0E DB 0E            0
0E DC 1A            LDAXD
0E DD 77            MOVMA
0E DE 13            INXD
0E DF 23            INXH
0E E0 1A            LDAXD
0E E1 77            MOVMA
0E E2 13            INXD
0E E3 23            INXH
0E E4 C9            RET
```

```
0E E5 CD  LASTPG, CALL
0E E6 9A          ERROUT  /TYPE OUT THE # OF ERRORS
0E E7 0F          0
0E E8 3A          LDA     /THEN PRINT ENOUGH CRS AND LFS TO
0E E9 38          PAGCNT
0E EA 18          0
0E EB 3D          DCRA    /FINISH THE PAGE
0E EC 4F          MOVCA
0E ED CD  MORLP,  CALL    /PRINT A CR & LF
0E EE 39          CRLF
0E EF 02          0
0E F0 0D          DCRC    /PRINT ENOUGH YET ?
0E F1 C2          JNZ     /NO, TYPE ANOTHER SET
0E F2 ED          MORLP
0E F3 0E          0
0E F4 CD          CALL    /YES, END OF THE PAGE
0E F5 84          SIXSPC  /TYPE 6 ADDITIONAL CRS & LFS
0E F6 0F          0
0E F7 CD          CALL
0E F8 90          DASH    /THEN TYPE 6 DASHES
0E F9 0F          0
0E FA C3          JMP
0E FB 84          SIXSPC  /THEN FINALLY, 6 MORE CRS & LFS
0E FC 0F          0

0E FD 3A  FINPAG, LDA     /GET THE NUMBER OF LINES PRINTED
0E FE 38          PAGCNT  /SO FAR.
0E FF 18          0
0F 00 FE          CPI     /NEW PAGE?
0F 01 36          066
0F 02 C8          RZ
0F 03 4F          MOVCA   /NO, GET THE NUMBER OF
0F 04 CD  NXTCR,  CALL    /LINES LEFT OVER.
0F 05 39          CRLF
0F 06 02          0
0F 07 0D          DCRC
0F 08 C2          JNZ
0F 09 04          NXTCR
0F 0A 0F          0
0F 0B C3          JMP     /AFTER FINISHING THE PAGE, TYPE
0F 0C 3B          PAGE1   /THE DASHES AND THE TITLE ON THE
0F 0D 0F          0       /NEXT PAGE

          /THIS IS JUST USED FOR TYPING OUT ERROR MESSAGES
          /PUSHH AND POPH BEFORE AND AFTER ENTRY IF THEY ARE USEFUL

0F 0E CD  LIST1,  CALL    /ENTER WITH H&L POINTING
0F 0F 39          CRLF    /TO THE ERROR MESSAGE
0F 10 02          0
0F 11 CD          CALL
0F 12 7A          TOUT    /PRINT THE ERROR MESSAGE
0F 13 05          0
0F 14 21          LXIH
0F 15 2C          LNER    /THEN PRINT "AT LINE #"
```

```
0F 16 12          0
0F 17 CD          CALL
0F 18 7A          TOUT      /THEN PRINT THE BLOCK NUMBER AND LINE
0F 19 05          0         /NUMBER WHERE THE ERROR OCCURRED
0F 1A 21          LXIH
0F 1B 2D          THOU+2    /POINT TO HUN. THOU (BLOCK COUNTER)
0F 1C 18          0
0F 1D 0E          MVIC
0F 1E 02          002
0F 1F CD          CALL
0F 20 1C          NPRINT    /TYPE OUT THE BLOCK # (00-99)
0F 21 06          0
0F 22 3E          MVIA
0F 23 2D          "-"       /NOW TYPE A DASH
0F 24 CD          CALL
0F 25 0F          IOO       /I-O DEVICE, OUTPUT TYPE
0F 26 18          0
0F 27 0E          MVIC
0F 28 04          004       /H&L POINT TO MSB OF LINE COUNTER !
0F 29 CD          CALL      /JUST SET THE # CNTR TO 4
0F 2A 1C          NPRINT    /AND PRINT THE LINE NUMBER
0F 2B 06          0
0F 2C 21          LXIH      /NOW INCREMENT THE
0F 2D 48          NMBERR    /NUMBER OF ERRORS COUNTER
0F 2E 18          0
0F 2F 34          INRM
0F 30 C3          JMP       /AND THEN PRINT A CR, LF
0F 31 39          CRLF
0F 32 02          0

          /"PG" IS RESPONSIBLE FOR DIVIDING THE ASSEMBLED
          /PRINTOUT INTO PAGES, TYPING THE TITLE AND PAGE
          /NUMBER.

0F 33 3A  PG,     LDA       /GET THE COUNTER FOR THE # OF
0F 34 38          PAGCNT    /CR & LF'S TYPED ON THIS PAGE SO FAR.
0F 35 18          0
0F 36 3D          DCRA      /DECREMENT AND SAVE THE NEW COUNT
0F 37 32          STA
0F 38 38          PAGCNT
0F 39 18          0
0F 3A C0          RNZ       /RETURN IF COUNT IS NOT 0.
0F 3B CD  PAGE1,  CALL      /COUNT IS 0, START A NEW PAGE
0F 3C 84          SIXSPC    /TYPE 6 CR,LF'S TO COMPLETE THIS PAGE
0F 3D 0F          0
0F 3E CD  PAGE2,  CALL
0F 3F 90          DASH      /THEN TYPE A LINE WITH 6 DASHES ON IT.
0F 40 0F          0
0F 41 0E          MVIC      /NOW TYPE ONLY 2 CR & LF'S
0F 42 02          002       /AFTER THE DASHED LINE.
0F 43 CD          CALL
0F 44 86          SIXSPC+2
0F 45 0F          0
0F 46 0E          MVIC      /NOW PRINT 6 SPACES FROM THE LEFT MARGIN
```

```
0F 47 06            006
0F 48 3E            MVIA
0F 49 A0            240
0F 4A CD            CALL
0F 4B 88            SIXSPC+4
0F 4C 0F            0
0F 4D E5            PUSHH    /SAVE THE TEXT POINTER
0F 4E 21            LXIH     /AND SET H&L TO POINT TO THE
0F 4F 9B            TYASS    /MESSAGE "TYCHON EDITOR-ASSEMBLER"
0F 50 11            0
0F 51 CD            CALL     /NOW PRINT THE MESSAGE
0F 52 7A            TOUT
0F 53 05            0
0F 54 0E  SPCOUT,   MVIC     /THEN TYPE 24 SPACES OVER
0F 55 18            030
0F 56 3E            MVIA
0F 57 A0            240
0F 58 CD            CALL
0F 59 88            SIXSPC+4
0F 5A 0F            0
0F 5B 23            INXH
0F 5C CD            CALL     /PAGE (#)
0F 5D 7A            TOUT
0F 5E 05            0
0F 5F 21            LXIH     /NOW TYPE THE BLOCK NUMBER
0F 60 2D            THOU+2
0F 61 18            0
0F 62 0E            MVIC
0F 63 02            002      /2 DIGITS TO TYPE
0F 64 CD            CALL
0F 65 1C            NPRINT
0F 66 06            0
0F 67 3E            MVIA
0F 68 2D            "-"
0F 69 CD            CALL
0F 6A 0F            IOO
0F 6B 18            0
0F 6C 3A            LDA      /GET THE PAGE NUMBER
0F 6D 47            PAGNMB
0F 6E 18            0
0F 6F 3C            INRA     /INCREMENT IT
0F 70 32            STA      /STORE IT
0F 71 47            PAGNMB
0F 72 18            0
0F 73 CD            CALL     /AND NOW PRINT IT
0F 74 50            DECOUT
0F 75 10            0
0F 76 CD            CALL
0F 77 39            CRLF
0F 78 02            0
0F 79 E1            POPH
0F 7A 3E            MVIA
0F 7B 36            066
0F 7C 32            STA
```

```
0F 7D 38            PAGCNT
0F 7E 18            0
0F 7F 0E            MVIC
0F 80 02            002
0F 81 C3            JMP
0F 82 86            SIXSPC+2
0F 83 0F            0
0F 84 0E   SIXSPC,  MVIC     /PRINT 6 LINE FEEDS.
0F 85 06            006
0F 86 3E            MVIA
0F 87 8A            212
0F 88 CD            CALL
0F 89 0F            100
0F 8A 18            0
0F 8B 0D            DCRC
0F 3C C2            JNZ
0F 8D 88            SIXSPC+4
0F 8E 0F            0
0F 8F C9            RET

0F 90 0E   DASH,    MVIC     /PRINT SIX DASHES (------).
0F 91 06            006
0F 92 3E            MVIA
0F 93 AD            255
0F 94 CD            CALL
0F 95 88            SIXSPC+4
0F 96 0F            0
0F 97 C3            JMP
0F 98 39            CRLF
0F 99 02            0

0F 9A CD   ERROUT,  CALL     /TYPE OUT THE # OF ERRORS
0F 9B 39            CRLF     /THAT OCCURRED AS WE ASSEMBLED THE
0F 9C 02            0        /PROGRAAM.
0F 9D 21            LXIH
0F 9E 7B            ERROR
0F 9F 11            0
0F A0 CD            CALL
0F A1 7A            TOUT
0F A2 05            0
0F A3 3A            LDA
0F A4 48            NMBERR
0F A5 18            0
0F A6 C3            JMP
0F A7 50            DECOUT
0F A8 10            0

           /"BYTE" IS USED TO BREAK UP THE 8 BIT
           /MNEMONIC, DATA OR ADDRESS VALUE
           /INTO A 2 BIT AND A 6 BIT WORD.

0F A9 3A   BYTE,    LDA      /"PORL" IS USED TO STORE THE ANSWER
0F AA 41            PORL     /TO THE QUESTION "AUTO-LOAD" DURING
0F AB 18            0        /THE ASSEMBLY PROCESS.
```

```
0F AC FE            CPI      /AUTO-LOAD?
0F AD 59            "Y"
0F AE C2            JNZ      /NO, THEN JUST PUNCH THE VALUE
0F AF E1            NOAUTO   /ON THE "IOP" DEVICE.
0F B0 0F            0
0F B1 E5            PUSHH    /YES, SAVE SOME REGISTERS.
0F B2 D5            PUSHD
0F B3 C5            PUSHB
0F B4 11            LXID     /LOAD D&E WITH THE LOWEST ADDRESS
0F B5 00            START    /USED TO STORE THE TEA PROGRAM.
0F B6 00            0
0F B7 21            LXIH     /LOAD H&L WITH THE HIGHEST ADDRESS
0F B8 C2            END      /USED TO STORE THE TEA PROGRAM.
0F B9 14            0
0F BA CD            CALL     /NOW SEE IF "ADDCNT" IS BETWEEN
0F BB FA            CHKOL    /THESE TWO ADDRESSES.  IF SO, PRINT "OL"
0F BC 0F            0        /IF WE ARE OK, RETURN
0F BD 2A            LHLD     /NOW MAKE SURE THAT "ADDCNT" IS NOT
0F BE 22            TXTBUF   /BETWEEN "TXTBUF" AND "SYMADF".
0F BF 18            0
0F C0 EB            XCHG
0F C1 2A            LHLD     /"TXTBUF" IS THE BEGINNING OF THE
0F C2 3B            SYMADF   /TEXT BUFFER AND "SYMADF" IS THE LAST
0F C3 18            0        /ADDRESS USED BY THE SYMBOLIC ADDR. TAB.
0F C4 CD            CALL     /IF THERE IS OVERLAP, PRINT "OL"
0F C5 FA            CHKOL    /IF WE ARE OK, RETURN.  IF NOT,
0F C6 0F            0        /PRINT "OL."
0F C7 2A            LHLD     /NOW MAKE SURE THAT WE ARE NOT TRYING
0F C8 12            BYTAB    /SAVE THE VALUE IN MEMORY IN THE
0F C9 18            0        /"DEFINED-BYTE" TABLE
0F CA EB            XCHG     /WHICH GOES FROM "BYTAB" UP TO
0F CB 2A            LHLD     /"ENDRAM."
0F CC 24            ENDRAM
0F CD 18            0
0F CE CD            CALL     /IF THERE IS OVERLAP, PRINT "OL."
0F CF FA            CHKOL    /OTHERWISE, RETURN.
0F D0 0F            0
0F D1 11            LXID     /NOW MAKE SURE THAT WE ARE NOT SAVING
0F D2 00            IOTEST   /THE VALUE IN THE TEMPORARY STORAGE
0F D3 18            0        /AREA, STACK, ETC.
0F D4 21            LXIH
0F D5 28            BUFF2+200
0F D6 19            0
0F D7 CD            CALL     /IF THERE IS OVERLAP HERE, PRINT "OL."
0F D8 FA            CHKOL    /FINALLY, IF THERE IS NO OVERLAP,
0F D9 0F            0        /THE VALUE CAN BE SAVED IN MEMORY.
0F DA 3A            LDA      /GET THE EIGHT-BIT VALUE
0F DB 36            MNEVAL
0F DC 18            0
0F DD 77            MOVMA    /SAVE THE VALUE IN THE MEMORY LOCATION
0F DE C1            POPB     /ADDRESSED BY "ADDCNT."  H&L ARE LOADED
0F DF D1            POPD     /WITH THE CONTENT OF "ADDCNT" BY THE
0F E0 E1            POPH     /"CHKOL" SUBROUTINE.
0F E1 3A  NOAUTO,   LDA      /REGARDLESS OF WHETHER OR NOT THE
```

```
0F E2 36          MNEVAL  /VALUE WAS "AUTOLOADED," GET IT AGAIN
0F E3 18          0
0F E4 4F  BYTE1,  MOVCA   /SAVE IT IN C
0F E5 3A          LDA     /GET THE CURRENT CHECKSUM
0F E6 3F          CHKSUM
0F E7 18          0
0F E8 81          ADDC    /ADD THE VALUE TO BE PUNCHED
0F E9 32          STA     /AND SAVE THE NEW CHECKSUM
0F EA 3F          CHKSUM
0F EB 18          0
0F EC 79          MOVAC   /GET THE VALUE BACK
0F ED E6  BYTE2,  ANI     /PUNCH THE 2 MSBS
0F EE C0          300
0F EF 07          RLC
0F F0 07          RLC
0F F1 CD          CALL
0F F2 09          IOP     /I-O DEVICE, PUNCH TYPE
0F F3 18          0
0F F4 79          MOVAC   /THEN PUNCH THE 6 LSBS.
0F F5 E6          ANI
0F F6 3F          077     /1 2-BIT BYTE AND 1 6-BIT BYTE.
0F F7 C3          JMP
0F F8 09          IOP     /I-O DEVICE, PUNCH TYPE
0F F9 18          0

0F FA E5  CHKOL,  PUSHH   /PUT THE HI ADDRESS ON THE STACK
0F FB C1          POPB    /AND POP IT INTO B&C
0F FC 2A          LHLD    /LOAD REGISTER PAIR H WITH
0F FD 34          ADDCNT  /THE ADDRESS WERE THE BYTE IS
0F FE 18          0       /TO BE STORED
0F FF 7D          MOVAL
10 00 93          SUBE
10 01 7C          MOVAH
10 02 9A          SBBD    /IS ADDCNT < LO ADDRESS?
10 03 D8          RC      /YES, SO RETURN
10 04 79          MOVAC
10 05 95          SUBL
10 06 78          MOVAB
10 07 9C          SBBH,
10 08 D8          RC      /ADDCNT > HI ADDRESS
10 09 21  YESOL,  LXIH    /THERE IS OVERLAP, SO LOAD   REG-
10 0A 69          OVLAP   /ISTER PAIR H WITH THE STARTING
10 0B 11          0       /ADDRESS FOR THE ERROR MESSAGE "OL"
10 0C CD          CALL    /PRINT THE ERROR MESSAGE AND THE
10 0D 0E          LIST1   /LINE NUMBER WHERE THE OVERLAP
10 0E 0F          0       /OCCURRED.
10 0F AF          XRAA    /THEN DISABLE THE AUTO-LOADER
10 10 32          STA     /FEATURE, SO THAT WE ONLY GET ONE
10 11 41          PORL    /ERROR MESSAGE, RATHER THAN A
10 12 18          0       /BUNCH OF THEM.
10 13 C1          POPB
10 14 D1          POPD
10 15 E1          POPH
10 16 33          INXSP   /DESTROY THE RETURN ADDRESS FOR
```

```
10 17 33          INXSP   /THE "CHKOL" SUBROUTINE.
10 18 C3          JMP     /AND PUNCH THE CONTENT OF
10 19 E1          NOAUTO  /"MNEVAL" ON THE IOP DEVICE.
10 1A 0F          0

          /"SPCIT" PRINTS A DIFFERENT NUMBER OF SPACES DURING
          /THE THIRD PASS, DEPENDING ON WHETHER HEX OR OCTAL
          /NUMBERS ARE PRINTED FOR ADDRESS AND DATA VALUES.

10 1B 0E  SPCIT,  MVIC
10 1C 0D          015
10 1D 3A          LDA
10 1E 3E          HORO
10 1F 18          0
10 20 FE          CPI
10 21 48          "H"
10 22 C2          JNZ
10 23 27          OSPC
10 24 10          0
10 25 0E          MVIC
10 26 0A          012
10 27 3E  OSPC,   MVIA
10 28 A0          240
10 29 C3          JMP
10 2A 88          SIXSPC+4
10 2B 0F          0

          /"UD" IS A COMMON ERROR, SO THIS SUBROUTINE PRINTS
          /THE UD ERROR MESSAGE FOLLOWED BY THE LINE NUMBER.

10 2C E5  UDOUT,  PUSHH
10 2D 21          LXIH
10 2E 91          ERROR2
10 2F 11          0
10 30 CD          CALL
10 31 0E          LIST1
10 32 0F          0
10 33 E1          POPH
10 34 C9          RET

          /"CLRREG" SETS SOME COUNTERS AND TEMPORARY STORAGE
          /LOCATIONS TO ZERO BEFORE EACH PASS OF THE ASSEMBLER.

10 35 AF  CLRREG, XRAA
10 36 32          STA     /SET THE "DEFINED ADDRESS INDICATOR"
10 37 16          ADCHK   /TO ZERO.
10 38 18          0
10 39 32          STA     /SET THE CHECKSUM TO 000
10 3A 3F          CHKSUM
10 3B 18          0
10 3C 32          STA     /SET THE "CURRENT PAGE NUMBER" TO 000
10 3D 47          PAGNMB
10 3E 18          0
10 3F 32          STA     /CLEAR THE ERROR COUNTER TO 000
```

```
10 40 48          NMBERR
10 41 18          0
10 42 2E  CLRRGI, MVIL
10 43 00          000
10 44 26          MVIH
10 45 00          000
10 46 22          SHLD    /THEN SET THE "CURRENT ADDRESS COUNTER"
10 47 34          ADDCNT  /TO 000 000. THIS IS THE DEFAULT
10 48 18          0       /STARTING ADDRESS
10 49 22          SHLD
10 4A 32          LINCNT  /ALSO SET THE LINE COUNTER FOR
10 4B 18          0       /ERROR MESSAGES TO 0.
10 4C 2A          LHLD    /LOAD REGISTER PAIR H WITH THE STARTING
10 4D 22          TXTBUF  /ADDRESS OF THE TEXT BUFFER.
10 4E 18          0
10 4F C9          RET

          /"DECOUT" PRINTS THE CONTENT OF THE A REGISTER
          /ON THE "IOO" DEVICE IN THE FORM OF A THREE-DIGIT
          /DECIMAL NUMBER (0-255).

10 50 D5  DECOUT, PUSHD
10 51 E5          PUSHH
10 52 F5          PUSHPSW
10 53 AF          XRAA
10 54 4F          MOVCA
10 55 57          MOVDA
10 56 5F          MOVEA
10 57 F1          POPPSW
10 58 47          MOVBA
10 59 D6  HUN,    SUI
10 5A 64          144
10 5B DA          JC
10 5C 62          TEN
10 5D 10          0
10 5E 1C          INRE
10 5F C3          JMP
10 60 59          HUN
10 61 10          0
10 62 C6  TEN,    ADI
10 63 64          144
10 64 D6          SUI
10 65 0A          012
10 66 DA          JC
10 67 6D          TINU
10 68 10          0
10 69 14          INRD
10 6A C3          JMP
10 6B 64          TEN+2
10 6C 10          0
10 6D C6  TINU,   ADI
10 6E 0A          012
10 6F 4F          MOVCA
10 70 7B          MOVAE
```

```
10 71 CD          CALL
10 72 24          BCDOUT
10 73 04          0
10 74 7A  NOHUN,  MOVAD
10 75 CD          CALL
10 76 24          BCDOUT
10 77 04          0
10 78 79  NOTEN,  MOVAC
10 79 CD          CALL
10 7A 24          BCDOUT
10 7B 04          0
10 7C E1          POPH
10 7D D1          POPD
10 7E C9          RET
```

```
                    /THESE ARE THE ROUTINES THAT THE
                    /PSEUDO-OPS "DW" AND "DB" USE.

10 7F CD  DW,       CALL     /ONLY IF ITS PASS 2 (PASS=120=P)
10 80 F3            PASSCK   /DO WE ENTER THE DEFINED WORD
10 81 10            0        /INTO THE SYMBOLIC ADDRESS TABLE.
10 82 CD            CALL     /ITS PASS 2, FIND THE CHARACTER
10 83 0A            FNDSYM   /STRING AFTER THE "DW"
10 84 11            0
10 85 EB            XCHG     /NOW SEE IF ITS IN THE
10 86 06            MVIB     /SYMBOLIC ADDRESS TABLE ALREADY
10 87 03            003
10 88 CD            CALL
10 89 32            CHKTB1   /CHECK THE TABLE
10 8A 11            0
10 8B C2            JNZ      /ITS IN THE TABLE ALREADY
10 8C 23            PSEUDO   /NOTHING ELSE TO DO.
10 8D 11            0
10 8E 2A  SYMENT,   LHLD     /OK, ENTER THE DEFINED WORD
10 8F 45            TMPADD   /THIS WAS SET BY "SEARCH"
10 90 18            0
10 91 C3            JMP      /NOW SAVE THE SYMBOL
10 92 02            NOSYM+3  /AND THE ADDRESS
10 93 0B            0

10 94 CD  DB,       CALL     /ONLY IF ITS PASS 2 (PASS=120=P)
10 95 F3            PASSCK   /DO WE ENTER THE DEFINED BYTE
10 96 10            0        /INTO THE "BYTE TABLE"; "BYTAB"
10 97 CD            CALL     /ITS PASS 2, FIND THE CHARACTER
10 98 0A            FNDSYM   /STRING AFTER THE "DB"
10 99 11            0
10 9A EB            XCHG
10 9B CD            CALL     /IS IT IN THE "BYTAB" ALREADY ?
10 9C 47            CHKTB3
10 9D 11            0
10 9E C2            JNZ      /YES, PROCESS THE NEXT LINE OF CODE
10 9F 23            PSEUDO
10 A0 11            0
10 A1 2A  DBOK,     LHLD     /ITS NOT IN THE TABLE, SO
10 A2 45            TMPADD   /ENTER IT (TMPADD SET BY "SEARCH")
10 A3 18            0
10 A4 06            MVIB     /NOW COUNT HOW MANY CHARACTERS
10 A5 00            000
10 A6 7E  DBOK1,    MOVAM    /WILL BE ENTERED INTO THE TABLE
10 A7 FE            CPI
10 A8 20            040      /WE MUST HAVE A SPACE AFTER THE STRING
10 A9 CA            JZ
10 AA B1            ENDDB    /DB AST 012
10 AB 10            0
10 AC 23            INXH     /INCREMENT THE POINTER
10 AD 04            INRB .   /AND THE COUNT
10 AE C3            JMP
10 AF A6            DBOK1
10 B0 10            0
```

```
10 B1 04  ENDDB,  INRB    /MAKE SPACE FOR THE = SIGN AND
10 B2 04          INRB    /NUMERIC VALUE
10 B3 2A          LHLD
10 B4 12          BYTAB   /GET THE LAST ADDRESS USED FOR
10 B5 18          0       /THE "BYTE TABLE"
10 B6 7D          MOVAL
10 B7 90          SUBB
10 B8 6F          MOVLA
10 B9 D2          JNC     /NO CARRY, DON'T DECREMENT THE
10 BA BD          HOKNOW  /H PART OF THE ADDRESS
10 BB 10          0
10 BC 25          DCRH    /UNDERFLOW, SO DECREMENT H BY 1
10 BD EB  HOKNOW, XCHG
10 BE 2A          LHLD    /NOW SEE IF ITS GROWN DOWN INTO THE
10 BF 3B          SYMADF  /END OF THE SYMBOLIC ADDRESS TABLE
10 C0 18          0
10 C1 7B          MOVAE   /D&E ARE THE "BYTAB" SO THEY SHOULD
10 C2 95          SUBL    /BIGGER THAN H&L OF "SYMADF"
10 C3 47          MOVBA
10 C4 7A          MOVAD
10 C5 9C          SBBH
10 C6 DA          JC      /NO, THE TABLE BOUNDARIES HAVE CROSSED
10 C7 23          MM      /TYPE THE "ME" - MEMORY EXCEEDED
10 C8 0B          0       /ERROR MESSAGE.
10 C9 EB          XCHG    /H&L ARE NOW "BYTAB"
10 CA 22          SHLD
10 CB 12          BYTAB   /SAVE THE NEW, LOWER ADDRESS FOR
10 CC 18          0       /THE NEXT ENTRY INTO "BYTAB"
10 CD EB          XCHG    /D&E ARE "BYTAB"
10 CE 2A          LHLD    /H&L POINT TO THE BEGINNING OF THE STRING
10 CF 45          TMPADD
10 D0 18          0
10 D1 7E  DBIN,   MOVAM   /GET A TEXT BUFFER CHARACTER AND
10 D2 FE          CPI     /STORE IT IN THE "BYTAB"
10 D3 20          040     /FIND THE SPACE TERMINATOR ?
10 D4 CA          JZ      /YES, THEN SAVE THE = AND THE NUMERIC
10 D5 DD          ADDEQ   /VALUE OF THE DEFINED BYTE
10 D6 10          0
10 D7 12          STAXD   /NOT A SPACE, SAVE IT IN THE TABLE
10 D8 23          INXH    /INCREMENT BOTH MEMORY POINTERS
10 D9 13          INXD
10 DA C3          JMP     /AND TRY ANOTHER CHARACTER
10 DB D1          DBIN
10 DC 10          0
10 DD 3E  ADDEQ,  MVIA    /WE FOUND THE SPACE TERMINATOR
10 DE 3D          075     /SO SAVE AN = SIGN IN THE TABLE
10 DF 12          STAXD
10 E0 D5          PUSHD   /SAVE THE "BYTAB" POINTER
10 E1 CD          CALL    /NOW GET THE NUMERIC VALUE FOR
10 E2 27          THRNMB  /THE CHARACTER STRING
10 E3 0D          0
10 E4 C2          JNZ     /NUMBER WAS INTERPRETABLE
10 E5 EC          DBINOK
10 E6 10          0
```

```
10 E7 CD          CALL    /BAD NUMBER, PRINT AN ERROR MESSAGE
10 E8 0E          BNOUT
10 E9 0D          0
10 EA 0E          MVIC    /SET THE NUMBER TO 000
10 EB 00          000
10 EC D1  DBINOK, POPD
10 ED 13          INXD
10 EE 79          MOVAC
10 EF 12          STAXD   /SAVE THE VALUE IN THE TABLE
10 F0 C3          JMP
10 F1 23          PSEUDO  /AND GO TO THE NEXT LINE
10 F2 11          0

          /DETERMINE THE CURRENT PASS #.  IF IT IS NOT
          /PASS 2, WE EXIT TO "PSEUDO" AND CONTINUE THE
          /ASSEMBLY PROCESS.  IF IT IS PASS 2, WE
          /RETURN TO THE CALLING ROUTINE.

10 F3 3A  PASSCK, LDA
10 F4 42          PASS    /GET THE PASS NUMBER
10 F5 18          0
10 F6 FE          CPI
10 F7 50          "P"     /IF ITS "P", ITS PASS 2
10 F8 C2          JNZ     /ITS NOT PASS TWO, SO RETURN
10 F9 06          PASSEX
10 FA 11          0
10 FB 3A          LDA     /IT IS PASS TWO, IS THE USER TRYING
10 FC 16          ADCHK   /TO USE THE DB OR DW PSEUDO-OP AFTER
10 FD 18          0       /AN ADDRESS HAS BEEN DEFINED BY AN *?
10 FE B7          ORAA
10 FF C8          RZ      /ITS OK, NO ADDRESS HAS BEEN DEFINED
11 00 21          LXIH    /NO GOOD, USING DB OR DW AFTER A
11 01 75          DEFER   /*, SO PRINT THE "DE" - DEFINITION
11 02 11          0       /ERROR MESSAGE
11 03 C3          JMP     /THIS IS A FATAL ERROR
11 04 26          FATAL
11 05 0B          0

11 06 E1  PASSEX, POPH    /DESTROY THE RETURN ADDRESS
11 07 C3          JMP     /OF THE CALLING ROUTINE AND CONTINUE
11 08 23          PSEUDO  /THE ASSEMBLY PROCESS
11 09 11          0

11 0A 2A  FNDSYM, LHLD    /FIND THE FIRST CHARACTER AFTER
11 0B 30          TFMLIN  /THE "DB" OR "DW"
11 0C 18          0
11 0D 23          INXH    /GET PAST THE "DB" OR "DW"
11 0E 23  FS1,    INXH
11 0F 7E          MOVAM   /SPACE TERMINATOR YET ?
11 10 FE          CPI
11 11 20          040
11 12 CA          JZ
11 13 0E          FS1
11 14 11          0
```

```
11 15 FE            CPI
11 16 0A            LF
11 17 DA            JC
11 18 0E            FS1       /TABS BETWEEN THE DB OR DW AND THE STRING
11 19 11            0
11 1A FE            CPI
11 1B 0D            CR        /WE FOUND A CARRIAGE RETURN BUT NO STRING
11 1C C2            JNZ
11 1D 2C            FSLS      /NO, SEE IF IT WAS A SLASH FOR A COMMENT
11 1E 11            0
11 1F E1   NOFSYM,  POPH      /DESTROY THE RETURN ADDRESS
11 20 CD            CALL      /FOR "FNDSYM" AND PRINT UD AT XX-NNNN
11 21 2C            UDOUT
11 22 10            0
11 23 CD   PSEUDO,  CALL      /SHOULD WE TYPE OUT THE PSEUDO-OP ?
11 24 E4            SPECL1
11 25 08            0
11 26 CD   PS1,     CALL
11 27 C7            NODATA
11 28 0D            0         /FIND THE NEXT LINE OF CODE
11 29 C3            JMP
11 2A 89            GENENT    /AND JUMP TO THE GENERAL ENTRY POINT
11 2B 09            0
11 2C FE   FSLS,    CPI
11 2D 2F            "/"       /SLASH FOR A COMMENT ?
11 2E CA            JZ
11 2F 1F            NOFSYM    /YES, SO WHERE'S THE STRING ?
11 30 11            0         /THERE IS NONE, SO ERROR
11 31 C9            RET       /OK, WE MUST HAVE H&L POINTING
                              /TO THE BEGINNING OF THE STRING

11 32 2A   CHKTB1,  LHLD      /THIS IS CALLED BY THE "DW" ROUTINE
11 33 39            SYMADD
11 34 18            0
11 35 0E   CHKTB2,  MVIC
11 36 3D            075       /SEARCH TERMINATOR
11 37 CD            CALL
11 38 6E            SEARCH    /SEARCH THE TABLE FOR AN
11 39 0B            0         /IDENTICAL ENTRY
11 3A FE            CPI
11 3B 9B            BTERM
11 3C C8            RZ        /FIND THE END OF THE TABLE ?, YES
11 3D 21            LXIH      /NO, THEN THERE IS ALREADY AN ENTRY
11 3E 94            REDEF     /WITH THE SAME NAME, SO ERROR
11 3F 11            0         /"REDEF"
11 40 CD            CALL
11 41 0E            LIST1
11 42 0F            0
11 43 AF            XRAA      /SO WE CAN JNZ TO ANY ERROR ROUTINES
11 44 FE            CPI
11 45 FF            377
11 46 C9            RET
11 47 2A   CHKTB3,  LHLD      /THIS IS FOR THE "BYTAB" SEARCH USED
11 48 12            BYTAB     /WHEN WE HAVE A "DB" PSEUDO-OP
```

```
11 49 18          0
11 4A 06          MVIB
11 4B 02          002
11 4C C3          JMP
11 4D 35          CHKTB2
11 4E 11          0
```

```
11 4F 8D  TAS,     215
11 50 8A           212
11 51 CD           315     /M OR T ?
11 52 A0           240
11 53 CF           317
11 54 D2           322
11 55 A0           240
11 56 D4           324
11 57 A0           240
11 58 BF           277
11 59 A0           240
11 5A 00           000

11 5B 8D  AUTO,    215
11 5C 8A           212
11 5D 41           101
11 5E 55           125     /AUTO-LOAD?
11 5F 54           124
11 60 4F           117
11 61 2D           055
11 62 4C           114
11 63 4F           117
11 64 41           101
11 65 44           104
11 66 A0           240
11 67 BF           277
11 68 00           000

11 69 4F  OVLAP,   117     /OVER-LAP (USED DURING AUTO-LOAD)
11 6A 4C           114
11 6B 00           000

11 6C CE  NZERO,   316     /NO ZERO
11 6D DA           332
11 6E 00           000

11 6F C2  BN,      302     /BAD NUMBER
11 70 CE           316
11 71 00           000

11 72 CD  ME,      315     /MEMORY EXCEEDED
11 73 C5           305
11 74 00           000

11 75 44  DEFER,   104     /DE - DEFINITION (DB OR DW) ERROR
11 76 45           105
11 77 00           000

11 78 4E  NOADR,   116     /NA - NO (DEFINED; *) ADDRESS
11 79 41           101
11 7A 00           000

11 7B C5  ERROR,   305     /ERRORS DETECTED
11 7C D2           322
```

```
11 7D D2            322
11 7E CF            317
11 7F D2            322
11 80 D3            323
11 81 A0            240
11 82 C4            304
11 83 C5            305
11 84 D4            324
11 85 C5            305
11 86 C3            303
11 87 D4            324
11 88 C5            305
11 89 C4            304
11 8A A0            240
11 8B BD            275
11 8C A0            240

11 8D 00            000
11 8E 52   RDY,     122      /R?
11 8F BF            277
11 90 00            000

11 91 D5   ERROR2,  325      /UD=UNDEFINED
11 92 C4            304
11 93 00            000

11 94 D2   REDEF,   322      /RE-DEFINED
11 95 C5            305
11 96 C4            304
11 97 C5            305
11 98 C6            306
11 99 A0            240
11 9A 00            000

11 9B D4   TYASS,   324      /TYCHON EDITOR-ASSEMBLER
11 9C D9            331
11 9D C3            303
11 9E C8            310
11 9F CF            317
11 A0 CE            316
11 A1 A0            240
11 A2 C5            305
11 A3 C4            304
11 A4 C9            311
11 A5 D4            324
11 A6 CF            317
11 A7 D2            322
11 A8 AD            255
11 A9 C1            301
11 AA D3            323
11 AB D3            323
11 AC C5            305
11 AD CD            315
11 AE C2            302
```

```
11 AF CC          314
11 B0 C5          305
11 B1 D2          322
11 B2 A0          240
11 B3 D6          326
11 B4 AD          255
11 B5 B3          263
11 B6 00          000
11 B7 D0          320      /PAGE
11 B8 C1          301
11 B9 C7          307
11 BA C5          305
11 BB A0          240
11 BC 00          000

11 BD 41  ALLET,  101
11 BE 42          102
11 BF 43          103
11 C0 44          104
11 C1 45          105
11 C2 48          110      /H
11 C3 4C          114      /L
11 C4 4D          115      /M
11 C5 FF          377      /TERMINATOR

11 C6 07  ALLETV, 007
11 C7 00          000
11 C8 01          001
11 C9 02          002
11 CA 03          003
11 CB 04          004
11 CC 05          005
11 CD 06          006

11 CE C8  HEXSRC, 310      /HEXADECIMAL
11 CF C5          305
11 D0 D8          330
11 D1 C1          301
11 D2 C4          304
11 D3 C5          305
11 D4 C3          303
11 D5 C9          311
11 D6 CD          315
11 D7 C1          301
11 D8 CC          314
11 D9 00          000

11 DA CF  OCTSRC, 317      /OCTAL
11 DB C3          303
11 DC D4          324
11 DD C1          301
11 DE CC          314
11 DF 00          000
```

```
        /THIS TABLE CONTAINS ALL OF THE ASCII VALUES FOR THE
        /VALID COMMANDS.  THEY ARE IN AN ORDER DETERMINED BY
        /THE ORDER OF THE JUMP INSTRUCTIONS IN THE "CMDJMP"
        /TABLE.

11 E0 44  CMDTAB, 104     /D=DELETE
11 E1 58          130     /X=ASSEMBLE
11 E2 4C          114     /L=LIST
11 E3 53          123     /S=SEARCH
11 E4 50          120     /P=PUNCH
11 E5 51          121     /Q=QUERY
11 E6 4E          116     /N=NEXT
11 E7 48          110     /H=HEXADECIMAL
11 E8 4F          117     /O=OCTAL
11 E9 52          122     /R=READER
11 EA 41          101     /A=APPEND
11 EB 43          103     /C=CHANGE
11 EC 49          111     /I=INSERT
11 ED 45          105     /E=EXCHANGE
11 EE 4D          115     /M=MEMORY LIMITS
11 EF 5A          132     /Z=RETURN TO SYSTEM MONITOR
11 F0 42          102     /B=BOOTSTRAP
11 F1 00          000     /USER COMMAND
11 F2 00          000

11 F3 4D  MEAIC,  115     /ME1 (MEMORY EXCEEDED !)
11 F4 45          105
11 F5 21          041
11 F6 00          000

11 F7 8D  SCIM,   215     /SOURCE CURRENTLY IN MEMORY ?
11 F8 8A          212
11 F9 D3          323
11 FA CF          317
11 FB D5          325
11 FC D2          322
11 FD C3          303
11 FE C5          305
11 FF A0          240
12 00 C3          303
12 01 D5          325
12 02 D2          322
12 03 D2          322
12 04 C5          305
12 05 CE          316
12 06 D4          324
12 07 CC          314
12 08 D9          331
12 09 A0          240
12 0A C9          311
12 0B CE          316
12 0C A0          240
12 0D CD          315
12 0E C5          305
```

```
12 0F CD            315
12 10 CF            317
12 11 D2            322
12 12 D9            331
12 13 A0            240
12 14 BF            277
12 15 A0            240
12 16 00            000

12 17 C1  AYS,      301     /ARE YOU SURE ?
12 18 D2            322
12 19 C5            305
12 1A A0            240
12 1B D9            331
12 1C CF            317
12 1D D5            325
12 1E A0            240
12 1F D3            323
12 20 D5            325
12 21 D2            322
12 22 C5            305
12 23 A0            240
12 24 BF            277
12 25 A0            240
12 26 00            000

12 27 A0  AT,       240     / AT
12 28 41            101
12 29 54            124
12 2A A0            240
12 2B 00            000

12 2C A0  LNER,     240     /AT LINE #
12 2D C1            301
12 2E D4            324
12 2F A0            240
12 30 CC            314
12 31 C9            311
12 32 CE            316
12 33 C5            305
12 34 A0            240
12 35 A3            243
12 36 A0            240
12 37 00            000

12 38 3D  EQSPC,    075     /=
12 39 A0  EQUI,     240
12 3A A0            240
12 3B 00            000

12 3C 8D  IA,       215     /INITIAL ADDRESS ?
12 3D 8A            212
12 3E 49            111
12 3F 4E            116
```

```
12 40 49            111
12 41 54            124
12 42 49            111
12 43 41            101
12 44 4C            114
12 45 A0            240
12 46 41            101
12 47 44            104
12 48 44            104
12 49 52            122
12 4A 45            105
12 4B 53            123
12 4C 53            123
12 4D A0            240
12 4E BF            277
12 4F A0            240
12 50 00            000

12 51 8D  FA,       215     /FINAL ADDRESS ?
12 52 8A            212
12 53 46            106
12 54 49            111
12 55 4E            116
12 56 41            101
12 57 4C            114
12 58 A0            240
12 59 41            101
12 5A 44            104
12 5B 44            104
12 5C 52            122
12 5D 45            105
12 5E 53            123
12 5F 53            123
12 60 A0            240
12 61 BF            277
12 62 A0            240
12 63 00            000

12 64 8D  CEQ,      215     /C=
12 65 8A            212
12 66 43            103
12 67 3D            075
12 68 00            000

12 69 42  BDCHKS,   102     /BAD CHECKSUM!
12 6A 41            101
12 6B 44            104
12 6C A0            240
12 6D 43            103
12 6E 48            110
12 6F 45            105
12 70 43            103
12 71 4B            113
12 72 53            123
```

```
12 73 55            125
12 74 4D            115
12 75 21            041
12 76 00            000

          /THIS TABLE CONTAINS ALL OF THE VALID 8080 MNEMONICS

12 77 4A  MNELST,   112      /JMP
12 78 4D            115
12 79 50            120
12 7A 3D            075
12 7B C3            303
12 7C 43            103      /CALL
12 7D 41            101
12 7E 4C            114
12 7F 4C            114
12 80 3D            075
12 81 CD            315
12 82 4A            112      /JC
12 83 43            103
12 84 3D            075
12 85 DA            332
12 86 4A            112      /JNC
12 87 4E            116
12 88 43            103
12 89 3D            075
12 8A D2            322
12 8B 4A            112      /JZ
12 8C 5A            132
12 8D 3D            075
12 8E CA            312
12 8F 4A            112      /JNZ
12 90 4E            116
12 91 5A            132
12 92 3D            075
12 93 C2            302
12 94 4A            112      /JP
12 95 50            120
12 96 3D            075
12 97 F2            362
12 98 4A            112      /JM
12 99 4D            115
12 9A 3D            075
12 9B FA            372
12 9C 4A            112      /JPE
12 9D 50            120
12 9E 45            105
12 9F 3D            075
12 A0 EA            352
12 A1 4A            112      /JPO
12 A2 50            120
12 A3 4F            117
12 A4 3D            075
12 A5 E2            342
```

```
12 A6 43          103     /CC
12 A7 43          103
12 A8 3D          075
12 A9 DC          334
12 AA 43          103     /CNC
12 AB 4E          116
12 AC 43          103
12 AD 3D          075
12 AE D4          324
12 AF 43          103     /CZ
12 B0 5A          132
12 B1 3D          075
12 B2 CC          314
12 B3 43          103     /CNZ
12 B4 4E          116
12 B5 5A          132
12 B6 3D          075
12 B7 C4          304
12 B8 43          103     /CP
12 B9 50          120
12 BA 3D          075
12 BB F4          364
12 BC 43          103     /CM
12 BD 4D          115
12 BE 3D          075
12 BF FC          374
12 C0 43          103     /CPE
12 C1 50          120
12 C2 45          105
12 C3 3D          075
12 C4 EC          354
12 C5 43          103     /CPO
12 C6 50          120
12 C7 4F          117
12 C8 3D          075
12 C9 E4          344
12 CA 4C          114     //LXIB
12 CB 58          130
12 CC 49          111
12 CD 42          102
12 CE 3D          075
12 CF 01          001
12 D0 4C          114     /LXID
12 D1 58          130
12 D2 49          111
12 D3 44          104
12 D4 3D          075
12 D5 11          021
12 D6 4C          114     /LXIH
12 D7 58          130
12 D8 49          111
12 D9 48          110
12 DA 3D          075
12 DB 21          041
```

```
12 DC 4C        114     /LXISP
12 DD 58        130
12 DE 49        111
12 DF 53        123
12 E0 50        120
12 E1 3D        075
12 E2 31        061
12 E3 53        123     /STA
12 E4 54        124
12 E5 41        101
12 E6 3D        075
12 E7 32        062
12 E8 4C        114     /LDA
12 E9 44        104
12 EA 41        101
12 EB 3D        075
12 EC 3A        072
12 ED 53        123     /SHLD
12 EE 48        110
12 EF 4C        114
12 F0 44        104
12 F1 3D        075
12 F2 22        042
12 F3 4C        114     /LHLD
12 F4 48        110
12 F5 4C        114
12 F6 44        104
12 F7 3D        075
12 F8 2A        052
12 F9 4E        116     /NOP
12 FA 4F        117
12 FB 50        120
12 FC 3D        075
12 FD 00        000
12 FE 53        123     /STAXB
12 FF 54        124
13 00 41        101
13 01 58        130
13 02 42        102
13 03 3D        075
13 04 02        002
13 05 49        111     /INXB
13 06 4E        116
13 07 58        130
13 08 42        102
13 09 3D        075
13 0A 03        003
13 0B 52        122     /RLC
13 0C 4C        114
13 0D 43        103
13 0E 3D        075
13 0F 07        007
13 10 44        104     /DADB
13 11 41        101
```

```
13 12 44          104
13 13 42          102
13 14 3D          075
13 15 09          011
13 16 4C          114      /LDAXB
13 17 44          104
13 18 41          101
13 19 58          130
13 1A 42          102
13 1B 3D          075
13 1C 0A          012
13 1D 44          104      /DCXB
13 1E 43          103
13 1F 58          130
13 20 42          102
13 21 3D          075
13 22 0B          013
13 23 52          122      /RRC
13 24 52          122
13 25 43          103
13 26 3D          075
13 27 0F          017
13 28 53          123      /STAXD
13 29 54          124
13 2A 41          101
13 2B 58          130
13 2C 44          104
13 2D 3D          075
13 2E 12          022
13 2F 49          111      /INXD
13 30 4E          116
13 31 58          130
13 32 44          104
13 33 3D          075
13 34 13          023
13 35 52          122      /RAL
13 36 41          101
13 37 4C          114
13 38 3D          075
13 39 17          027
13 3A 44          104      /DADD
13 3B 41          101
13 3C 44          104
13 3D 44          104
13 3E 3D          075
13 3F 19          031
13 40 4C          114      /LDAXD
13 41 44          104
13 42 41          101
13 43 58          130
13 44 44          104
13 45 3D          075
13 46 1A          032
13 47 44          104      /DCXD
```

```
13 48 43          103
13 49 58          130
13 4A 44          104
13 4B 3D          075
13 4C 1B          033
13 4D 52          122      /RAR
13 4E 41          101
13 4F 52          122
13 50 3D          075
13 51 1F          037
13 52 49          111      /INXH
13 53 4E          116
13 54 58          130
13 55 48          110
13 56 3D          075
13 57 23          043
13 58 44          104      /DAA
13 59 41          101
13 5A 41          101
13 5B 3D          075
13 5C 27          047
13 5D 44          104      /DADH
13 5E 41          101
13 5F 44          104
13 60 48          110
13 61 3D          075
13 62 29          051
13 63 44          104      /DCXH
13 64 43          103
13 65 58          130
13 66 48          110
13 67 3D          075
13 68 2B          053
13 69 43          103      /CMA
13 6A 4D          115
13 6B 41          101
13 6C 3D          075
13 6D 2F          057
13 6E 49          111      /INXSP
13 6F 4E          116
13 70 58          130
13 71 53          123
13 72 50          120
13 73 3D          075
13 74 33          063
13 75 53          123      /STC
13 76 54          124
13 77 43          103
13 78 3D          075
13 79 37          067
13 7A 44          104      /DADSP
13 7B 41          101
13 7C 44          104
13 7D 53          123
```

```
13 7E 50        120
13 7F 3D        075
13 80 39        071
13 81 44        104     /DCXSP
13 82 43        103
13 83 58        130
13 84 53        123
13 85 50        120
13 86 3D        075
13 87 3B        073
13 88 43        103     /CMC
13 89 4D        115
13 8A 43        103
13 8B 3D        075
13 8C 3F        077
13 8D 48        110     /HLT
13 8E 4C        114
13 8F 54        124
13 90 3D        075
13 91 76        166
13 92 52        122     /RNZ
13 93 4E        116
13 94 5A        132
13 95 3D        075
13 96 C0        300
13 97 50        120     /POPB
13 98 4F        117
13 99 50        120
13 9A 42        102
13 9B 3D        075
13 9C C1        301
13 9D 50        120     /PUSHB
13 9E 55        125
13 9F 53        123
13 A0 48        110
13 A1 42        102
13 A2 3D        075
13 A3 C5        305
13 A4 41        101     /ADI
13 A5 44        104
13 A6 49        111
13 A7 3D        075
13 A8 C6        306
13 A9 52        122     /RST0
13 AA 53        123
13 AB 54        124
13 AC 30        060
13 AD 3D        075
13 AE C7        307
13 AF 52        122     /RZ
13 B0 5A        132
13 B1 3D        075
13 B2 C8        310
13 B3 52        122     /RET
```

```
13 B4 45        105
13 B5 54        124
13 B6 3D        075
13 B7 C9        311
13 B8 41        101     /ACI
13 B9 43        103
13 BA 49        111
13 BB 3D        075
13 BC CE        316
13 BD 52        122     /RST1
13 BE 53        123
13 BF 54        124
13 C0 31        061
13 C1 3D        075
13 C2 CF        317
13 C3 52        122     /RNC
13 C4 4E        116
13 C5 43        103
13 C6 3D        075
13 C7 D0        320
13 C8 50        120     /POPD
13 C9 4F        117
13 CA 50        120
13 CB 44        104
13 CC 3D        075
13 CD D1        321
13 CE 4F        117     /OUT
13 CF 55        125
13 D0 54        124
13 D1 3D        075
13 D2 D3        323
13 D3 50        120     /PUSHD
13 D4 55        125
13 D5 53        123
13 D6 48        110
13 D7 44        104
13 D8 3D        075
13 D9 D5        325
13 DA 53        123     /SUI
13 DB 55        125
13 DC 49        111
13 DD 3D        075
13 DE D6        326
13 DF 52        122     /RST2
13 E0 53        123
13 E1 54        124
13 E2 32        062
13 E3 3D        075
13 E4 D7        327
13 E5 52        122     /RC
13 E6 43        103
13 E7 3D        075
13 E8 D8        330
13 E9 49        111     /IN
```

```
13 EA 4E        116
13 EB 3D        075
13 EC DB        333
13 ED 53        123     /SBI
13 EE 42        102
13 EF 49        111
13 F0 3D        075
13 F1 DE        336
13 F2 52        122     /RST3
13 F3 53        123
13 F4 54        124
13 F5 33        063
13 F6 3D        075
13 F7 DF        337
13 F8 52        122     /RPO
13 F9 50        120
13 FA 4F        117
13 FB 3D        075
13 FC E0        340
13 FD 50        120     /POPH
13 FE 4F        117
13 FF 50        120
14 00 48        110
14 01 3D        075
14 02 E1        341
14 03 58        130     /XTHL
14 04 54        124
14 05 48        110
14 06 4C        114
14 07 3D        075
14 08 E3        343
14 09 50        120     /PUSHH
14 0A 55        125
14 0B 53        123
14 0C 48        110
14 0D 48        110
14 0E 3D        075
14 0F E5        345
14 10 41        101     /ANI
14 11 4E        116
14 12 49        111
14 13 3D        075
14 14 E6        346
14 15 52        122     /RST4
14 16 53        123
14 17 54        124
14 18 34        064
14 19 3D        075
14 1A E7        347
14 1B 52        122     /RPE
14 1C 50        120
14 1D 45        105
14 1E 3D        075
14 1F E8        350
```

```
14 20 50          120     /PCHL
14 21 43          103
14 22 48          110
14 23 4C          114
14 24 3D          075
14 25 E9          351
14 26 58          130     /XCHG
14 27 43          103
14 28 48          110
14 29 47          107
14 2A 3D          075
14 2B EB          353
14 2C 58          130     /XRI
14 2D 52          122
14 2E 49          111
14 2F 3D          075
14 30 EE          356
14 31 52          122     /RST5
14 32 53          123
14 33 54          124
14 34 35          065
14 35 3D          075
14 36 EF          357
14 37 52          122     /RP
14 38 50          120
14 39 3D          075
14 3A F0          360
14 3B 50          120     /POPPSW
14 3C 4F          117
14 3D 50          120
14 3E 50          120
14 3F 53          123
14 40 57          127
14 41 3D          075
14 42 F1          361
14 43 44          104     /DI
14 44 49          111
14 45 3D          075
14 46 F3          363
14 47 50          120     /PUSHPSW
14 48 55          125
14 49 53          123
14 4A 48          110
14 4B 50          120
14 4C 53          123
14 4D 57          127
14 4E 3D          075
14 4F F5          365
14 50 4F          117     /ORI
14 51 52          122
14 52 49          111
14 53 3D          075
14 54 F6          366
14 55 52          122     /RST6
```

```
14 56 53          123
14 57 54          124
14 58 36          066
14 59 3D          075
14 5A F7          367
14 5B 52          122      /RM
14 5C 4D          115
14 5D 3D          075
14 5E F8          370
14 5F 53          123      /SPHL
14 60 50          120
14 61 48          110
14 62 4C          114
14 63 3D          075
14 64 F9          371
14 65 45          105      /EI
14 66 49          111
14 67 3D          075
14 68 FB          373
14 69 43          103      /CPI
14 6A 50          120
14 6B 49          111
14 6C 3D          075
14 6D FE          376
14 6E 52          122      /RST7
14 6F 53          123
14 70 54          124
14 71 37          067
14 72 3D          075
14 73 FF          377
14 74 44          104      /"DB" PSEUDO-OP
14 75 42          102
14 76 3D          075
14 77 08          010
14 78 44          104      /"DW" PSEUDO-OP
14 79 57          127
14 7A 3D          075
14 7B 10          020
14 7C 52          122      /RIM
14 7D 49          111
14 7E 4D          115
14 7F 3D          075
14 80 20          040
14 81 53          123
14 82 49          111      /SIM
14 83 4D          115
14 84 3D          075
14 85 30          060

                  /THE "PARTIAL" MNEMONICS FOLLOW
                  /THE ASSEMBLER COMBINES THE
                  /PARTIAL MNEMONIC VALUE WITH THE
                  /REGISTER DESIGNATION TO PRODUCE
                  /THE 8 BIT OCTAL VALUE.
```

```
14 86 4D          115     /MVI
14 87 56          126
14 88 49          111
14 89 3D          075
14 8A 06          006
14 8B 41          101     /ADD
14 8C 44          104
14 8D 44          104
14 8E 3D          075
14 8F 80          200
14 90 41          101     /ADC
14 91 44          104
14 92 43          103
14 93 3D          075
14 94 88          210
14 95 53          123     /SUB
14 96 55          125
14 97 42          102
14 98 3D          075
14 99 90          220
14 9A 53          123     /SBB
14 9B 42          102
14 9C 42          102
14 9D 3D          075
14 9E 98          230
14 9F 41          101     /ANA
14 A0 4E          116
14 A1 41          101
14 A2 3D          075
14 A3 A0          240
14 A4 58          130     /XRA
14 A5 52          122
14 A6 41          101
14 A7 3D          075
14 A8 A8          250
14 A9 4F          117     /ORA
14 AA 52          122
14 AB 41          101
14 AC 3D          075
14 AD B0          260
14 AE 43          103     /CMP
14 AF 4D          115
14 B0 50          120
14 B1 3D          075
14 B2 B8          270
14 B3 49          111     /INR
14 B4 4E          116
14 B5 52          122
14 B6 3D          075
14 B7 04          004
14 B8 44          104     /DCR
14 B9 43          103
14 BA 52          122
```

```
14 BB 3D          075
14 BC 05          005
14 BD 4D          115      /MOV
14 BE 4F          117
14 BF 56          126
14 C0 3D          075
14 C1 40          100
14 C2 9B  END,    233
```

```
AD2OR3 =0D C0  ADBIAS =0B E4  ADBYTE =0C C4  ADCHK  =18 16
ADCHK1 =00 D4  ADD1   =07 8C  ADDCHK =07 92  ADDCNT =18 34
ADDEQ  =10 DD  ADDINC =0D AB  ADDOK  =0C DC  ADIFF  =18 1B
ADRIN  =06 EB  AEND   =01 09  ALLBCK =01 91  ALLET  =11 BD
ALLETV =11 C6  ALLIST =02 AE  APPEND =01 03  ASSEMB =07 B7
ASSMB1 =07 CE  AT     =12 27  AUTO   =11 5B  AYS    =12 17
BACK1  =01 F2  BACK2  =01 F4  BADHL  =0A 0E  BADNMB =0D 9B
BBYTE  =07 9A  BBYTE1 =07 9D  BCDOUT =04 24  BCMDE  =00 F0
BDBIAS =0B F6  BDCHKS =12 69  BF1    =04 68  BFHUN  =18 4C
BFLINC =04 5D  BFOUT  =04 73  BFOUT1 =04 7B  BFTEN  =18 4B
BFTHOU =18 4D  BFUNIT =18 4A  BLKD   =03 44  BLKD1  =03 47
BLKLST =02 A8  BLKOR1 =04 0E  BLKT   =18 2D  BLKU   =18 2C
BN     =11 6F  BNOUT  =0D 0E  BNXTIN =07 69  BOOT   =07 52
BSORT  =0E 5F  BSORT1 =0E 65  BSORT2 =0E 76  BUFF2  =18 A8
BYTAB  =18 12  BYTE   =0F A9  BYTE1  =0F E4  BYTE2  =0F ED
CAT    =05 9A  CEQ    =12 64  CHANGE =05 6B  CHKFRM =07 5C
CHKIT  =06 26  CHKOL  =0F FA  CHKSUM =18 3F  CHKSYM =09 FB
CHKTB1 =11 32  CHKTB2 =11 35  CHKTB3 =11 47  CHNG1  =03 71
CLEAN  =06 2C  CLNAE  =04 2C  CLNALL =04 29  CLRREG =10 35
CLRRG1 =10 42  CMDJMP =00 9D  CMDOK  =00 9B  CMDTAB =11 E0
CNTCHR =03 8B  CNTCRS =02 F9  CNTLN  =01 26  COMDEC =00 80
CONS1  =04 A3  CONSER =05 20  CONVHX =0D 2E  COUT   =05 48
CR     =01 CB  CRFND  =0C 49  CRIO   =0D A0  CRLF   =02 39
CROK   =0A C6  DAIES  =06 C5  DASH   =0F 90  DB     =10 94
DBIN   =10 D1  DBINOK =10 EC  DBOK   =10 A1  DBOK1  =10 A6
DECOUT =10 50  DEFER  =11 75  DELALL =00 65  DELET  =03 20
DELIT  =03 4D  DELSTR =06 CC  DEONE  =03 3E  DEST   =0A 53
DIFF   =02 90  DIFF1  =03 63  DOIT   =08 5E  DONRDR =02 69
DPLS   =09 B6  DW     =10 7F  E1     =01 DA  ELINE  =01 4A
END    =14 C2  ENDBUF =01 1E  ENDCNT =03 96  ENDDB  =10 B1
ENDH   =18 25  ENDLN  =08 D5  ENDLST =04 DC  ENDPL  =02 FC
ENDPOP =03 5F  ENDRAM =18 24  ENDTXT =0C C1  ENTEST =04 DF
EOUT   =0E 32  EQSPC  =12 38  EQU1   =12 39  ERASE  =01 D5
ERROR  =11 7B  ERROR2 =11 91  ERROUT =0F 9A  ESA    =0B 2F
EVERY  =04 21  EXCHG  =06 9F  EXOK   =18 20  EXSTR  =18 64
FA     =12 51  FADRIN =00 2D  FATAL  =0B 26  FCR    =05 2D
FINADD =0C DF  FINDCR =0D A5  FINDIT =04 84  FINDM  =08 84
FINLN  =18 19  FINPAG =0E FD  FMNOK  =08 F0  FNDSYM =11 0A
FNL    =0A F9  FNXSYM =0E CD  FOUND  =09 3F  FRSTLN =02 C6
FS1    =11 0E  FSLS   =11 2C  FT     =08 92  GENENT =09 89
GETCHR =00 76  GETOND =01 2C  HEX09  =07 43  HEXIN  =07 26
HEXNMB =0D 46  HEXOCT =00 FC  HEXP   =07 CB  HEXSRC =11 CE
HIADD  =18 14  HLMOUT =0C 01  HLOUT  =05 AD  HOKNOW =10 BD
HORO   =18 3E  HUN    =10 59  IA     =12 3C  IADRIN =00 1B
INCHK  =06 51  INITLN =18 17  INOK   =03 E5  INPUT  =03 C2
INPUT1 =03 C7  INSERT =03 6B  INSIT  =03 B5  INSIT1 =03 B8
INSSTR =06 D8  IOCHK  =0C 12  IODONE =0D 01  IOECHO =18 06
IONOEC =18 03  IOO    =18 0F  IOP    =18 09  IOR    =18 0C
IORDR  =06 56  IOTAB  =07 A5  IOTEST =18 00  IOTRAN =00 10
LASTPG =0E E5  LCHAR  =02 D6  LDR    =05 55  LDR1   =05 57
LDRIN  =07 54  LDROK  =02 B8  LETOK  =0A A2  LINC   =05 F4
LINC1  =05 F7  LINCNT =18 32  LINE1  =06 08  LINE12 =06 0B
LIST   =02 75  LIST1  =0F 0E  LISTER =0C 20  LISTNT =0C 28
LNER   =12 2C  LNNMB  =06 17  LNNMB1 =06 1A  LOOK   =0A AB
```

```
LOOK1   =0A B4  LORP    =02 7D  LOWOK   =09 C9  LXI     =09 AD
MATH    =0A 85  MATH1   =0A 74  ME      =11 72  MEAIC   =11 F3
MEMOUT  =05 85  MM      =0B 23  MMEXC   =06 96  MNELST  =12 77
MNEND   =0B B0  MNEOK   =08 6D  MNEVAL  =18 36  MOK     =07 EC
MORCHR  =04 3D  MORLP   =0E ED  MORMNE  =08 3E  MORT    =18 3D
MOVE    =0A 70  MOVES   =0E D3  MOVI    =0A 65  MTCHR   =18 49
NCHAR   =0E 8D  NCRYET  =0D D5  NENTRY  =04 E4  NEWLIN  =04 E8
NEWTAB  =01 58  NM      =08 73  NMBERR  =18 48  NMBOK   =03 F8
NMONLY  =09 D2  NOA1    =0D 97  NOADR   =11 78  NOAUTO  =0F E1
NOBEG   =01 EB  NOCOMA  =08 C2  NOCOUT  =08 DC  NOCR    =0A E1
NOCRY   =0E 09  NOCT    =02 0D  NODATA  =0D C7  NOEC    =01 89
NOEQU   =0E 79  NOEQY   =0E 21  NOET    =02 16  NOET1   =02 17
NOFSYM  =11 1F  NOGOAD  =0D 06  NOGOOD  =03 EF  NOHUN   =10 74
NOMATH  =09 26  NONMB   =0D 82  NOOUT   =02 1A  NOPAGE  =0C A2
NOPART  =09 20  NOR     =0C 60  NOR1    =0C 69  NOREG   =0A A5
NOSHOW  =01 B8  NOSLSH  =08 B8  NOSR    =0A D7  NOSS    =0A CC
NOSTAR  =08 AE  NOSYM   =0A FF  NOSYMA  =0A 18  NOTA    =07 6F
NOTAP   =08 07  NOTAP1  =08 0A  NOTCC   =0C A5  NOTEN   =10 78
NOTEND  =0E 9A  NOTERM  =05 01  NOTPOM  =0B 8D  NOTSUM  =07 84
NOTY    =00 60  NPRINT  =06 1C  NSCIN   =04 CA  NTADD   =0C F6
NXCHR1  =08 97  NXTCHK  =04 EB  NXTCHR  =08 96  NXTCIN  =01 5D
NXTCMD  =00 89  NXTCR   =0F 04  NXTFND  =04 A0  NXTHEX  =07 2A
NXTIN   =03 77  NXTLET  =01 5A  NXTLOC  =05 4B  NXTMNE  =0B 87
NXTNBM  =05 F9  NXTNMB  =0D 67  NXTOCT  =07 03  NXTSK   =04 39
NZ      =0A 4B  NZER    =0A 3A  NZERO   =11 6C  OB      =09 D5
OCT1    =05 D8  OCTEND  =09 F7  OCTIN   =06 FF  OCTOUT  =05 B2
OCTQ    =09 8F  OCTSRC  =11 DA  OKCR    =05 4F  OKINC   =0D CA
OKQ2    =09 A9  ONEIN   =04 1C  ONELN   =01 3C  ONELST  =02 A5
ONEOK   =05 09  OPENIT  =03 AB  OSPC    =10 27  OVLAP   =11 69
PAGCNT  =18 38  PAGE1   =0F 3B  PAGE2   =0F 3E  PAGNMB  =18 47
PASS    =18 42  PASS1   =08 1C  PASS2   =08 33  PASS3   =08 53
PASSCK  =10 F3  PASSEX  =11 06  PCRLF   =05 61  PG      =0F 33
PHEX    =05 CE  POPMOR  =03 53  PORL    =18 41  PORM    =0B D4
POS     =07 C8  PS1     =11 26  PSEUDO  =11 23  QUEST   =00 6B
QUEST1  =00 70  QUOTE   =09 9B  RADD1   =0D 1F  RDLINE  =0C 7B
RDRIN   =02 43  RDRMOR  =0C 7E  RDY     =11 8E  READIT  =0D E2
READOK  =00 F5  READY   =0C 4F  REDEF   =11 94  REENTR  =0C 3C
REG     =18 43  RET2    =02 EF  RORK    =18 40  SAVADD  =0B 3C
SAVBIS  =0B E8  SAVBUF  =01 12  SAVMOR  =0B 06  SAVRC   =02 5F
SAVVAL  =0B A9  SCHAR   =02 D4  SCIM    =11 F7  SEARCH  =0B 6E
SECT    =03 F3  SEEK    =04 32  SEEK1   =04 4F  SENSE   =18 18
SETPL   =02 7A  SETTAB  =0C 26  SETTFC  =0D DC  SHOW    =01 97
SHOWIT  =01 AC  SHOWN   =01 C3  SHOWT   =01 C1  SIXSPC  =0F 84
SKIP    =0B 8E  SKPHIN  =07 2D  SKPMOR  =0B 72  SMNXT   =05 0E
SORT    =0D F3  SPC     =07 1D  SPCAFT  =05 EF  SPCIT   =10 1B
SPCOIN  =07 06  SPCOUT  =0F 54  SPECDF  =0B 4A  SPECL   =0C 23
SPECL1  =08 E4  SRCADD  =18 1D  SRCH1   =0B 79  STACK   =18 A0
START   =00 00  START1  =00 18  STOBUF  =03 86  STREND  =0B 97
STRFND  =18 1F  STRIN   =04 C8  STRING  =18 4E  SUBIAS  =0B ED
SWAP    =0E B0  SYMADD  =18 39  SYMADF  =18 3B  SYMCEQ  =0E AB
SYMCMA  =0A EB  SYMENT  =10 8E  SYMIN   =0A F7  SYML    =08 67
SYMOCT  =09 BC  SYMOUT  =0D EB  TAB     =02 25  TAB1    =02 F3
TABBR   =02 2D  TABOK   =0C 40  TABX    =02 07  TAKEDF  =02 97
TAPADD  =0D 17  TAPE    =02 70  TAPE1   =05 3D  TAS     =11 4F
```

```
TBLIM   =19 7F  TCBCHR =0A 93  TEN     =18 62  TERM    =04 01
TFCLIN  =18 2E  TFMLIN =18 30  THOU    =18 2B  THRNMB  =0D 27
TINU    =10 6D  TMP    =18 26  TMPADD  =18 45  TOUT    =05 7A
TRLR    =03 15  TRYOCT =0D 60  TTYI    =06 6A  TTYIN   =06 76
TTYOUT  =06 79  TWOHEX =07 22  TXTBUF  =18 22  TYASS   =11 9B
TYPEQ   =0E 2B  TYPSYM =0E 0B  UDOUT   =10 2C  UNIT    =18 28
VALLET  =0A 8B  VALTRM =0B 59  YESNO   =00 49  YESOL   =10 09
ZEDUT   =0A 45  ZEDUT1 =0A 48  ZEROIT  =06 0F  ZNEXT   =0A 1D

ERRORS DETECTED = 000
```

Cod. 322 P

31

TEA

Un Editor-Assembler Residente per

8080-8085

Christopher A. Titus

GRUPPO EDITORIALE JACKSON